WRDC-TR-89-1134

E&V GUIDEBOOK, VERSION 2.0

Bard S. Crawford
Peter G. Clark

The Analytic Sciences Corp.
55 Walker's Brook Drive
Reading, MA 01856

October 1989

Interim Report for Period Aug 88 to Sep 89

AVIONICS LABORATORY
WRIGHT RESEARCH DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6543

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the office of Public Affairs, (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public including foreign nations.

This technical report has been reviewed and is approved for publication.

Raymond Szymanski

Raymond Szymanski
Program Manager

3 October 1989
Date

FOR THE COMMANDER:

Charles H Krueger J

CHARLES H. KRUEGER JR
Director, System Avionics Division
Avionics Laboratory

05 OCT 1989
Date

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization, please notify WRDC/AAAF-3 WPAFB, OH 45433 to help us maintain a current mailing list.

Copies of this report should not be returned unless required by security considerations, contractual obligation, or notice on a specific document.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distriubution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TASC No. TR-5234-4	5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-89-1134
6a. NAME OF PERFORMING ORGANIZATION The Analytic Sciences Corporation 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Avionics Laboratory (WRDC/AAAF) Wright Research Development Center
6c. ADDRESS (City, State, and ZIP Code) 55 Walker's Brook Drive Reading, MA 01856	7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, OH 45433-6543
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Ada Joint Program Office 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-85-C-1812
8c. ADDRESS (City, State, and ZIP Code) Room 3E114 (1211 S. Fern Street) The Pentagon Washington, D.C. 20301-3080	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 63756D 63226 / PROJECT NO. 2853 / TASK NO 01 / WORK UNIT ACCESSION NO. 01
11. TITLE (Include Security Classification) E & V Guidebook, Version 2.0	
12. PERSONAL AUTHOR(S) Crawford Bard S., Clark, Peter, G.	
13a. TYPE OF REPORT Interim Technical	13b. TIME COVERED FROM 16 Aug88 TO 30Sep89
14. DATE OF REPORT (Year, Month, Day) October 1989	15. PAGE COUNT 151
16. SUPPLEMENTARY NOTATION A companion document titled " E & V Reference Manual" is being released concurrently.	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
09	02		EVALUATION VALIDATION Ada Programming Support Environments (APSES)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Ada community, including government, industry, and academic personnel, needs the capability to assess APSEs (Ada Programming Support Environments) and their components and to determine their conformance to applicable standards (e.g., DoD-STD-1838, the CAIS standard). The technology required to fully satisfy this need is extensive and largely unavailable; it cannot be acquired by a single government-sponsored, professional society-sponsored, or private effort. The purpose of the APSE Evaluation and Validation (E&V) task is to provide a focal point for addressing the need by:

(1) Identifying and defining specific technology requirements,
(2) Developing selected elements of the required technology,
(3) Encouraging others to develop some elements, and
(4) Collecting information describing existing elements.

(See Reverse)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL RAYMOND SZYMANSKI	22b. TELEPHONE (Include Area Code) (513)255-3947 / 22c. OFFICE SYMBOL WRDC/AAAF-3

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

Block 19 Continued

This information will be made available to DoD components, other government agencies, industry and academia.

The purpose of the E & V Guidebook (this document) is to provide information that will help users to assess APSEs and APSE components by :

(1) Assisting in the selection of E&V procedures, the interpretation of results, and integration of analyses and results.

(2) Describing E&V procedures and techniques developed by the E&V task, and

(3) Assisting in the location of E&V procedures and techniques developed outside the E&V Task.

All E&V procedures and techniques found in the Guidebook are referenced by the indexes contained in the companion document called the E&V Reference Manual.

EXECUTIVE SUMMARY

The Ada community, including Government, industry, and academic personnel, needs the capability to assess APSEs (Ada Programming Support Environments) and their components, and to determine their conformance to applicable standards (e.g., DoD-STD-1838, the CAIS standard). The technology required to fully satisfy this need is extensive and largely unavailable; it cannot be acquired by a single government-sponsored, professional society-sponsored, or private effort. The purpose of the APSE Evaluation and Validation (E&V) Task is to provide a focal point for addressing the need by:

(1) Identifying and defining specific technology requirements;
(2) Developing selected elements of this technology;
(3) Encouraging others to develop additional elements; and
(4) Collecting information describing elements which already exist.

This information will be made available to DoD components, other government agencies, industry and academia.

The purpose of the E&V Guidebook (this document) is to provide information that will help users to assess APSEs and APSE components by:

(1) Assisting in the selection of E&V procedures, the interpretation of results, and integration of analyses and results;
(2) Describing E&V procedures and techniques developed by the E&V Task; and
(3) Assisting in the location of E&V procedures and techniques developed outside the E&V Task.

All E&V procedures and techniques found in the Guidebook are referenced by the indexes contained in the companion document called the E&V Reference Manual.

Chapters 1 through 4 provide a general introduction to the document and other background material. Chapter 5 and later chapters are "formal chapters" built around a standard format and formal grammar. Each of the formal chapters contains all the assessment procedures and techniques associated with a particular group of tools or toolsets to be assessed, such as Compilation System Assessors or Test System Assessors. The assessment procedures are described and in some instances can be applied directly from the information given in the Guidebook. In other cases, the user is directed to a primary reference for more information.

Yearly updates and extensions to this document are planned. Therefore, comments and suggestions are welcome. Please send comments electronically (preferred) to szymansk@ajpo.sei.cmu.edu, or by regular mail to Mr. Raymond Szymanski, WRDC/AAAF, Wright Patterson AFB, OH 45433-6543.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE CF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 PURPOSE OF GUIDEBOOK

This document is a product of the Ada Programming Support Environment (APSE) Evaluation and Validation (E&V) Task sponsored by the Ada Joint Program Office. It is one of a pair of companion documents known as the E&V Reference System, consisting of:

- E&V Reference Manual
- E&V Guidebook.

The subject of both documents is the assessment of APSEs and their components. Specific assessment techniques typically fall into one of two categories: evaluation (assessment of performance and quality) and validation (assessment of conformance to a standard).

The purpose of the Guidebook is to provide a collection of information to support a variety of E&V users in the following ways. It should help them:

- Gain an overall understanding of APSE assessment, in particular, the selection of appropriate E&V procedures, the interpretation of test results, and the integration of analyses and results.
- Apply the various E&V procedures and techniques developed under E&V Task sponsorship.
- Find the primary sources for those E&V procedures and techniques not developed by the E&V Task or not fully explained within the Guidebook (due to space or other constraints).

The Reference Manual includes many "pointers" to sections in the Guidebook and other documents which describe E&V techniques in much the same way that a card catalog does in a library. Figure 1.1-1 illustrates the relationship between the documents.

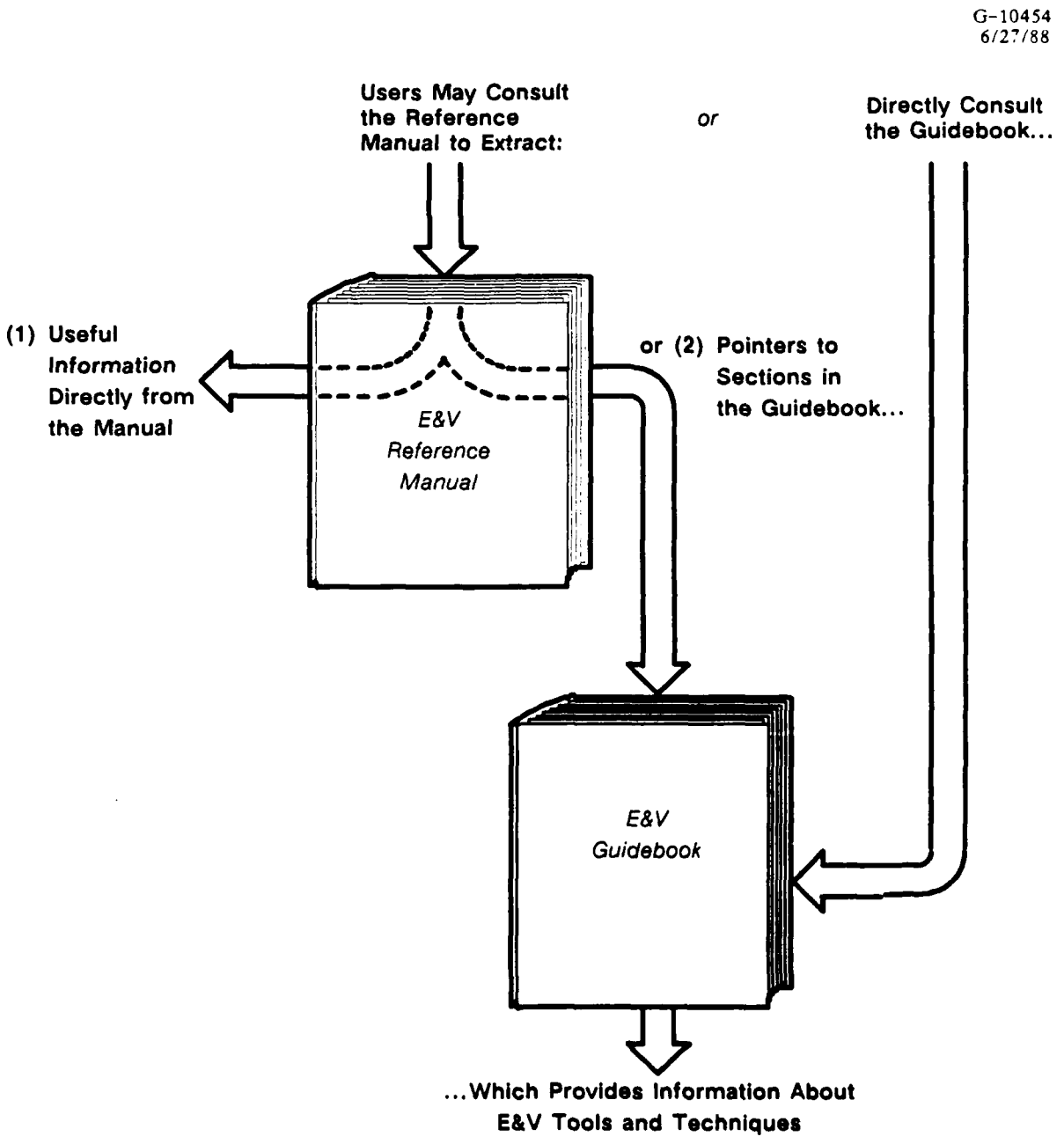

Figure 1.1-1 Relationship Between Reference Manual and Guidebook

1.2 THE NEED FOR E&V TECHNOLOGY

Technology for the assessment of APSEs and APSE components (tools) is needed because of the difficulty in assessing APSEs and because of the importance of the decisions made based on these assessments. The importance of an APSE selection is evident when one considers the large, critical, Ada-based systems to be developed in the coming years. The effectiveness, reliability, and cost of these systems will be strongly influenced by the environments used to develop and maintain them. From the point of view of a software developing organization, the decision to select an APSE can be an important investment decision with long-lasting influence on a number of projects and the organization's operating procedures, training, and competitiveness. From the point of view of a software maintenance organization, the environment used will strongly influence the organization's effectiveness, as well as the cost of its operations and training.

The difficulty of assessing APSEs and tools exists for several reasons. First, an APSE represents very complex technology with many elements, which can be assessed individually or in combination. Second there is a confusing diversity of choice with respect to individual tools, tool sets, or "whole APSEs"; and there are a number of ways of viewing APSEs; see Chapter 3 of the E&V Reference Manual [@RM 3].* Third, the state of the art of APSE architecture and of some categories of tools (e.g., graphic design tools) is undergoing rapid change. Finally, there is a lack of historical data relevant to APSEs, partly because of the general pace of technological change and partly because we are dealing with Ada, a relatively new implementation language. E&V technology provides methods and techniques to overcome these difficulties and provides a basis for determining performance and other attributes of APSEs.

In addition to the need for assessment technology itself, there is a need for information about this technology. Potential buyers and users of APSEs and tools need a framework for understanding APSEs and their assessment, as well as information about specific assessment techniques. Similarly, vendors of tools and APSEs need to be aware of the deficiencies of current products, as well as the criteria to be used in the assessment of future products. Such awareness on both sides, expressed in a common terminology, should speed up the evolution of better software development environments.

*The format used for references is associated with the "formal grammar" used beginning with Chapter 5. See further explanations in Appendix C.

1.3 BACKGROUND

In June 1983 the Ada Joint Program Office (AJPO) proposed the formation of the E&V Task and a tri-service E&V Team, with the Air Force designated as lead service. In October 1983 the Air Force officially accepted responsibility as lead service and designated the Air Force Wright Aeronautical Laboratories (AFWAL) at Wright Patterson Air Force Base as lead organization. In April 1984 an E&V Workshop was held at Airlie, Virginia. The purpose of the workshop was to solicit participation of industry representatives in the E&V Task. Many of the participants in the workshop have chosen to remain involved as Distinguished Reviewers, and additional industry participants have subsequently become involved in E&V Team activities.

The E&V Task publishes an annual public report. The following paragraph is quoted from the 1987 version [@E&V Report 1987] of the report:

> "The Ada community, including government, industry, and academic personnel, needs the capability to assess APSEs (Ada Programming Support Environments) and components and to determine their conformance to applicable standards (e.g., DoD-STD-1838, the CAIS standard). The technology required to fully satisfy this need is extensive and largely unavailable; it cannot be acquired by a single government-sponsored, professional society-sponsored, or private effort. The purpose of the APSE Evaluation and Validation (E&V) Task is to provide a focal point for addressing the need by (1) identifying and defining specific technology requirements, (2) developing selected elements of the required technology, (3) encouraging others to develop some elements, and (4) collecting information describing existing elements. This information will be made available to DoD components, other government agencies, industry, and academia."

The team public reports contain much additional information for the interested reader. See for example, the "DoD APSE Analysis Report" [@E&V Report 1984], the "Requirements for the Evaluation and Validation of Ada Programming Support Environments, Version 2.0" [@E&V Report 1987], and the "Tools and Aids Document, Version 1.0" [@E&V Report 1987], which are synopsized in Chapter 4 [4.3, 4.5, 4.6].

Three competitive contracts have been awarded under the E&V Task. These are:

- Technical Support contract - awarded June 1985
- Ada Compiler Evaluation Capability (ACEC) contract - awarded February 1987
- CAIS Implementation Validation Capability (CIVC) contract - awarded May 1987.

The major purpose of the first of these contracts is to create and update elements of the E&V Reference System, including this document. The purpose of the second and third contracts is to create two additional elements (ACEC and CIVC) of the needed E&V technology.

1.4 ORGANIZATION OF THE GUIDEBOOK

Chapter 2 provides a general description of the structure and use of the Guidebook.

Chapter 3 provides high-level guidance to users who may need assistance in selecting instances of the technology and integrating the results of its application.

Chapter 4 provides synopses of other documents or activities that are either too broad in scope to fit within one of the later chapters or are of historical importance to E&V activities.

Chapter 5 and subsequent chapters are "formal chapters" that describe or refer to specific instances of E&V technology. Each of the formal chapters contains all of the procedures and techniques associated with a particular group of tools or toolsets to be assessed, such as Compilation System Assessors or Test System Assessors. A standard format based on a "formal grammar" is used in presenting this material. See further explanations in Appendix C.

Appendices A, B, and C contain a list of citations, a list of acronyms and abbreviations, and a definition of the formal grammar used in the formal chapters, respectively. *Appendix D* contains the list of vendors and agents of assessment tools who are the primary sources of E&V technology.

2. STRUCTURE AND USE OF THE GUIDEBOOK

This chapter provides a brief explanation of the structure and uses of the E&V Guidebook. It is expected that many users have first consulted the E&V Reference Manual (see Fig. 1.1-1) and come to the Guidebook with a specific chapter and section number in hand, prepared to read about a specific instance of E&V technology. A user following this path does not particularly care about the overall structure of the document. Other users, however, may come to the document with a less narrowly-defined objective. An attempt has been made, with such users in mind, to make the Guidebook easy to use as a stand-alone document.

2.1 STRUCTURE

The Guidebook structure may be considered as having four major subdivisions, as follows:

- Introductory Material (Chapters 1 and 2)
- General Background Material (Chapters 3 and 4)
- Specific E&V Technology Descriptions (Chapters 5 and beyond)
- Appendices.

The introductory material is used to introduce the document and its structure. The general background material is used to introduce the general subject of APSE assessment. Chapter 3 is an "essay" designed to help users who are faced with the question of how to evaluate an APSE as a whole, or how to compare several APSEs with the objective of selecting one. (Chapter 3 of the E&V Reference Manual, dealing with whole APSE assessment issues, is a "companion essay" that provides complementary background material.) Chapter 4 provides a different kind of background material. It may be considered a "guide to the literature" of APSE assessment. It contains synopses of documents that fall into one of two categories. One category is that of documents that contain *no* specific instances of E&V technology, but contain generally useful background material. The other category is that of documents that contain or discuss *multiple*

instances of E&V technology, which are individually covered in multiple parts of the later, formal chapters. These multiple instances can be thought of as children of a common parent. In order to avoid the redundancy of summarizing the parent document many times, the Chapter 4 synopsis is provided as a common point to which all the children may refer. Each "synopsis text frame" in Chapter 4 has the following parts:

- Citation: (the primary reference)
- Synopsis: (brief description)
- Methods: (references to specific instances of E&V technology, if any).

The formal chapters (Chapters 5 and beyond), which comprise the main bulk of the Guidebook, describe or summarize specific instances of E&V technology. The chapter subjects and titles were chosen to be meaningful and intuitive to users of the Guidebook. Thus, they focus on the *subject* of assessment (e.g., Compilation System, Test System, Ada Design Support System, etc) rather than the *method* of assessment (e.g., formal validation, subjective evaluation, etc). Within each chapter there are, in general, multiple instances of assessment technology. Some may be examples of evaluation techniques, others may be examples of validations, others may be mixtures of the two. *Readers should not infer approval* of the E&V Task, because a tool or technique is included in the collection, *or disapproval*, because a tool or technique is not included. Readers who know of instances of E&V technology not reported here are urged to contact the E&V Task chairman, in the manner described in the Executive Summary. The separate instances within a chapter are simply placed there in *chronological order*, indicating the relative timing of the material's first appearance in the Guidebook. *Readers should not infer any judgment as to relative importance based on order.* Each chapter thus provides a dynamically growing section of the Guidebook. Old sections will not be thrown away or replaced by new sections describing newer techniques. Old sections may, however, be updated if a particular vendor or agent has updated material describing a technique, or has improved the technique itself without fundamentally changing the approach. Each "technique text frame" in the formal chapters has the following parts:

- Purpose:
- Primary References:

- Host/OS: (if applicable)
- Vendors/Agents: (if applicable)
- Method:

  Inputs:

  Process:

  Outputs:

The final major subdivision (the appendices) require little explanation here. The formal grammar described in Appendix C need not concern most users. It was employed because of the possibility of a future on-line, electronic version of the Reference System, supported by advanced updating and information retrieval techniques.

2.2 EXAMPLE USES

Instances of E&V technology may be found in two ways. A user may consult the Guidebook directly, or may first consult the E&V Reference Manual, as pictured in Figure 1.1-1. A user who comes directly to the Guidebook would typically first look at the Table of Contents. For example, a user interested in evaluating compiler performance would naturally look under Chapter 5 "Compilation System Assessors." The titles of Sections 5.2, "IDA Benchmarks," and 5.3, "Ada Compiler Evaluation Capability (ACEC)," would probably suggest themselves as relevant to this user's needs — as indeed they are.

Alternatively, the user may consult the E&V Reference Manual, which is designed to help find E&V techniques in the same way that the card catalog helps people find books in the library. For example, the Reference Manual contains both a Function Index and an Attribute Index, each of which contains cross references to elements in the other. One element of the Function Index is the function "Compilation," which is cross-referenced to a number of relevant attributes. Under the particular function-attribute pair "Compilation-Processing Effectiveness" are listed a number of Guidebook references. Among these are the same two Sections, 5.2 and 5.3, of the Guidebook, mentioned in the previous paragraph. The user following this procedure could pick up the Guidebook and go directly to these two sections or "text frames" and find sum-

mary information concerning the IDA Benchmarks test suite and the ACEC test suite, respectively.

2.3 BIAS IN EVALUATION

Some elements of bias are inherent in all evaluation techniques. Examples of such elements are given in the following paragraph. It is important that users of evaluation techniques be aware of these built-in biases and use caution in the interpretation of results. A tool or APSE that is "different" may receive an unfair evaluation because of an unintended bias against new technology or a new concept of operations. The effects of bias can be minimized, but not eliminated, by careful design of experiments. In some situations certain elements of bias are actually desirable, as discussed in the final paragraph of this section.

Consider, for example, a whole-APSE evaluation based on a series of structured experiments involving various portions of the life cycle. The items to be evaluated are competing commercial software products — collections of tools integrated in some way. The experiments are built around a model project, partially completed, and instructions to perform specific life-cycle activities such as test and integration, configuration management, response to a change in system requirements, or documentation updates. The outcome of such experiments are inevitably influenced by factors that are not characteristics of the software products under evaluation. These factors include: the skill and experience of the evaluation team members, the management ability of the team leaders, the software development methods ordinarily favored by the team members (as opposed to those best supported by the APSEs under evaluation), the application domain of the model project (as opposed to those in which team members are experienced), and other surrounding environmental factors.

The influence of the factors listed above can be controlled to some extent. For example, it is possible to employ a sequence in which: in Phase 1 Team 1 does Model Project M on APSE A while Team 2 does Model Project N on APSE B; in Phase 2 the teams exchange roles; in Phase 3 they compare notes and write a joint evaluation report. This sort of approach can be useful in removing bias, but is of course very expensive.

Before embarking on an APSE evaluation it is important to have a clear understanding of the purpose of the evaluation. It is unlikely, for example, that the purpose of an APSE evaluation project will be to select "the best APSE" in some general, global sense. It is more likely that the purpose will be to select the best APSE for a particular project or sequence of projects, to be used by a particular organization (with its unique history and preferences), in a particular application domain. It is also possible that there is only one APSE available or that an APSE has already been chosen. In this case the purpose of the evaluation includes obtaining a better understanding of the characteristics of the APSE and the risks and costs associated with its use in a particular application domain and with a particular development methodology. In such cases it is quite legitimate for the evaluation to reflect organizational and individual "biases." It is, after all, a particular group of individuals who will be asked to use the APSE in a productive way. If they have a choice, they will want to choose an APSE that supports their style. If the choice is already made, they will need to understand how the given APSE supports, or fails to support, their preferred methods of operation. Thus, a "biased evaluation" can be a desirable and necessary objective.

3. INTEGRATION OF APSE ASSESSMENTS

The purpose of this chapter is to provide high-level guidance for the user of the E&V Reference System (Reference Manual and Guidebook) who is interested in evaluating an APSE as a whole, or in comparing several APSEs with the objective of selecting one. While the "formal chapters" (beginning with Chapter 4 of the Reference Manual and Chapter 4 of the Guidebook) provide assistance in locating, defining, and assessing many individual aspects of APSEs, they do not provide an overall approach to weighting and combining the results of such assessments. Section 3.1 briefly discusses some relevant general background material. Section 3.2 discusses some earlier, partial efforts aimed at an integrated approach. Section 3.3 provides some additional guidance leading to a comprehensive, integrated approach.

It is necessary, first, to distinguish the subject of this chapter -- integrated whole-APSE assessment — from the subject of Chapter 13 — specific "Whole-APSE Assessors." The integrated form of whole-APSE assessment (Section 3.3) involves a combining or mixing together of the results of individual assessment steps to arrive at a decision. These individual steps may be oriented toward specific functions or tools, or may be oriented toward a "whole APSE," in relation to specific attributes or the APSE's performance in a specific life-cycle phase or activity. Thus, a whole APSE assessor (Chapter 13) might be used to evaluate the APSE's capability to support a project team during one major activity, such as preliminary design. The results of such an assessment would become one of the weighted factors of an integrating process leading to a major decision.

3.1 GENERAL BACKGROUND

Chapter 4 of this Guidebook contains synopses of books, articles, and documents. Some of these have historical value and are also indirectly relevant to the topic of an integrated approach to APSE evaluation because they provide definitions of an APSE or highlight issues that may be important during APSE evaluations. The Stoneman document [@DoD 1980] defines an APSE as a layered system and includes some discussion of evaluation criteria. The Common

APSE Interface Set (CAIS) definition documents [@DoD 1986, @MIL 1989] describe proposed interface requirements for interfaces that exist between layers of an APSE. The motivation for these interface requirements is to support the transportability of tools and project databases from one APSE to another. The book "Life Cycle Support in the Ada Environment" by McDermid and Ripken [@McDermid 1984] takes a top-down approach to defining a "coherent APSE," starting with requirements for a coherent life-cycle methodology; see synopsis [4.10]. Several papers in an IEEE Tutorial [@Wasserman 1981] provide relevant observations on desirable characteristics and major issues for Ada support environments; see synopses [4.11, 4.12, 4.13]. A more recent survey paper "Characteristics and Functions of Software Engineering Environments: An Overview" [@Houghton and Wallace 1987] provides a broad discussion of environments and the state of the art; see synopsis [4.15]. Chapter 3 of the E&V Reference Manual [@RM: Whole APSE Assessment Issues 3.] presents various ways of viewing an APSE and key whole-APSE attributes.

3.2 EARLY EFFORTS AT INTEGRATED APSE ASSESSMENT

The following quotation is from a paper by Henderson and Notkin [@Henderson 1987]:

> "Perhaps the biggest failing of environments research and development to date is the general lack of scientific evaluation of existing environments. Evaluation approaches and actual evaluations are beginning to appear, but relatively little effort has been given to this undeniably fundamental subject."

Some early efforts are mentioned briefly below.

The Software Engineering Institute (SEI) has developed a methodology [@Weiderman 1987] to evaluate certain aspects of APSEs. The methodology centers around the execution of several experiments in the environment(s) to be evaluated. The experiments are designed in a generic fashion and must be tailored or "instantiated" for each specific environment; see synopsis [4.13]. The early applications, by the SEI, of this methodology were aimed at limited objectives (see [7.2, 9.1, 10.2]), and are not examples of integrated whole-APSE assessments. However, the SEI methodology has been applied by TRW to a broad-gauged, whole-APSE selection process; see [@Gray 1987]. It is apparent that industry has devoted re-

sources internally to comparative assessment of commercial APSEs. However, other than Gray's paper, little has yet been published in the open literature describing the techniques employed.

The book "Selecting an Ada Environment" [@Lyons 1986], written by the Ada Europe Environment Working Group, provides background discussion about a broad range of topics. In each chapter and section it provides a list of questions to be asked about the environment under consideration; see synopsis [4.9]. Some of the chapters and questions listed have a definite "integrated whole-APSE assessment" flavor. The whole-APSE checklist [13.3] has been adapted from material of this book.

3.3 TOWARDS A COMPREHENSIVE APPROACH

The published literature on assessment of software engineering environments does not include descriptions of "decision support" oriented approaches. (But, see the forthcoming dissertation [@Lawlis 1989].) *A decision support system is one that leads a user through a* structured framework that includes weighting factors and decision criteria, and supports a final decision process. As applied to APSE assessment this kind of approach would support a final decision, such as, whether a single APSE under consideration is "good enough," or which of several APSEs under consideration is "best."

The following characteristics appear to be appropriate for a decision support system designed for integrated APSE assessment:

- The system should allow the specification of a list of "essential features" that are absolutely required for the contemplated application or family of applications. Ideally, each of these essential features would be subject to a question or test that yields an unambiguous "yes/no" result — yes, the required feature is present, or no, it is missing.
- The system should allow the specification of a second list of attributes and function-attribute pairs that represent desirable features or criteria, which should be involved in an integrated assessment.
- The system should allow for specification of "weights" to be applied to each attribute and function-attribute pair in the second list.

The weights will typically be chosen subjectively by the assessment participants.

- The system should include a mechanism to document/identify the method of assessment used for every test/metric to be employed in addressing every essential and desirable feature.
- The system should include a well-defined method of combination, leading to an overall set of pre-decision results. For example, the results may be summarized in two lines as in:

  1) satisfies all essential requirements (listed in Table A)

  2) scores 72 out of possible 100 (based on weights in Table B).

The characteristics outlined above represent a general framework that can be applied very differently by different users. At one extreme is a decision maker with little time or resources, who focuses on a short list of essential features only, and accepts answers supplied by vendors or vendor documentation. At the other extreme is a team of APSE assessors who conduct a comprehensive, detailed set of tests and "model project" experiments and expend multiple person-years of effort in a comparative, hands-on assessment of competing APSEs.

It is also possible that two assessment teams applying equal resources might differ greatly in the manner of their assessments. One might view the APSE as a support system for a particular life-cycle methodology adopted by its organization. Another might view the APSE as a project database management system. These two teams would be likely to use very different tests, or very different weights where the same tests are used. Neither is necessarily right or wrong. In the final analysis, it is the software developer's responsibility to understand his own application area and the most critical attributes of his development support environment.

4. SYNOPSES

The purpose of this chapter is to provide a single place in the Guidebook for synopses of documents (or other resources), which have too broad a scope to fit within one of the subsequent Chapters. In some cases the subject document appears only in this Chapter because it does *not* contain specific instances of E&V technology. For example, the Stoneman document [@DoD 1980] does not deal with evaluation or validation of APSEs, but it has general historical importance to the entire field of Ada environments and has been selected as the first document to be synopsized. In other cases a particular document may contain *multiple* instances of E&V technology, which are themselves summarized or referenced in multiple parts of the Guidebook. These multiple instances can be thought of as children of a common parent. In order to avoid the redundancy of summarizing the parent document many times, the Chapter 4 synopsis is provided as a common point to which all the children may refer. The formal grammar used to structure the entries in subsequent chapters includes, therefore, a mechanism for referring back to the synopsis contained in Chapter 4. Similarly, after each synopsis there is a provision for forward references to specific techniques (if any) described in later chapters.

Most of the documents synopsized in this chapter are readily available through public sources. A few of them may be difficult or impossible to obtain for some readers; these were included because the synopsis itself was judged to be helpful in filling in a piece of the historical background.

4.1 STONEMAN

Citations:

[DoD 1980] J.N. Buxton, "Requirements for Ada Programming Support Environments — STONEMAN," U.S. Department of Defense, February 1980, DTIC Number AD A100 404.

Synopsis:

The Stoneman document defines the APSE as a layered system. The innermost layer is referred to as the Kernel APSE, or KAPSE. The KAPSE is machine-dependent and includes the database functions and other general operating system support functions. The next layer, the Minimal APSE, or MAPSE, consists of the minimal set of tools which can support the development of software. The outermost layer, the APSE, consists of tools and functions that are project dependent. In addition to providing guidance for APSE designers, the Stoneman document provides some evaluation criteria for APSEs.

4.2 HOUGHTON: A TAXONOMY OF TOOL FEATURES FOR THE Ada PROGRAMMING SUPPORT ENVIRONMENT (APSE)

Citations:

[Houghton 1983] R.C., Houghton, Jr., "A Taxonomy of Tool Features for the Ada Programming Support Environment (APSE)," National Bureau of Standards, NBSIR-81-2625, February 1983.

Synopsis:

This paper puts forth a taxonomic classification of APSE features. The features included satisfy the criteria that they are "within current technology" and are "oriented to the Ada language." The top two levels of the classification are as follows:

- Input
  - Subject
  - Control Input
- Function
  - Transformation
  - Management
  - Static Analysis
  - Dynamic Analysis
- Output
  - User Output
  - Machine Output

For each of the second-level elements above, a third-level list is given, and some discussion is provided. The paper includes the results of a survey in which the second and third-level elements under "Function" are each rated as "Required," "Important," or "Useful."

4.3 E&V REPORT: DoD APSE ANALYSIS

Citations:

[E&V Report 1984] "DoD APSE Analysis Report, Draft Version 1.0," 31 August 1984, Appendix C of "Evaluation and Validation (E&V) Team Public Report", Air Force Wright Aeronautical Laboratories, November 1984, DTIC Number AD A153 609.

Synopsis:

The DoD Ada Programming Support Environment (APSE) Analysis Document was prepared by the APSE Working Group (APSEWG) of the E&V Team. It contains a description and analysis of the Ada programming support environments developed by each of the armed services. The three environments analyzed were the Air Force's Ada Integrated Environment (AIE), the Army's Ada Language System (ALS), and the Navy's Ada Language System/Navy (ALS/N). The design documentation was used to determine the functionality contained in each programming environment. The functions were described in a taxonomy in order to determine the commonality and differences of each system. The taxonomy developed for this purpose was an expanded version of the function part of the taxonomy developed earlier by Houghton [@Houghton 1983]; see synopsis [4.2].

4.4 CLASSIFICATION SCHEMA/E&V TAXONOMY CHECKLISTS

Citations:

[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

Synopsis:

The purpose of this document was to set forth a schema, or a framework, to be used in subsequent E&V documents, especially the E&V Reference Manual [@RM]. The Function Index of the schema was strongly influenced by earlier documents, such as Houghton's taxonomy [4.2], the DoD APSE Analysis Report [4.3], and the SEE tool features taxonomy [@Kean 1985]. The upper levels of the Function Index of the schema became the initial version of the Function Index of the Reference Manual. The lower levels were found to incorporate a large number of tool functions which could be evaluative in nature. These tool function features have been carried over into the Guidebook as capability assessment checklists. As a group, they are considered the Classification Schema Checklists.

Methods:

[Compilation Checklist	5.8;
Program Library Management Checklist	5.9;
Linking/Loading Checklist	6.2;
Import/Export Capabilities Checklist	6.3;
Debugging Capabilities Checklist	6.5;
Real-Time Analysis Capabilities Checklist	6.7;
Configuration Management Capabilities Checklist	10.1;
Electronic Mail Capabilities Checklist	15.3;
Text Editing Capabilities Checklist	99.1]

4.5 REQUIREMENTS FOR E&V

Citations:

[E&V Report 1987] "Requirements for Evaluation and Validation of Ada Programming Support Environments, Version 2.0," 4 December 1986, Appendix D of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, September 1987, DTIC Number AD A196 164.

[E&V Report 1984] "Requirements for Evaluation and Validation of Ada Programming Support Environments, Version 1.0," 17 October 1984, Appendix B of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, November 1984, DTIC Number AD A153 609.

Synopsis:

This document was prepared by the Requirements Working Group (REQWG) of the E&V Team. Its purpose is to set forth requirements on the E&V Task. It is intended for use by the E&V Team and by the E&V Task contractors in developing technology for the evaluation and validation of APSEs. However, its use in other E&V efforts is encouraged. The document contains three categories of requirements: (1) those on the E&V Team itself, (2) those on the E&V methods and procedures, and (3) those specifying what is to be evaluated or validated. See also the Tools and Aids Document, synopsis [4.6].

Version 1.0 of the document contains three questionnaires for assessing: command language interpreters, compilers, and configuration management tools.

Methods:

[Compiler Assessment Questionnaire 5.12;
Configuration Management Assessment Questionnaire 10.3;
Command Language Interpreter Assessment Questionnaire 99.4]

4.6 TOOLS AND AIDS FOR E&V

Citations:

[E&V Report 1987] "Tools and Aids Document, Version 1.0," September 1987, Appendix C of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, September 1987, DTIC Number AD A196 164.

Synopsis:

This document was prepared by the Requirements Working Group (REQWG) of the E&V Team. It identifies the community's E&V technology needs, provides definitions of those needs, and prioritizes them. The purpose of this document is to provide pertinent information to those agencies willing and able to fund the development of E&V technology. It reflects the E&V Requirements Document (see synopsis [4.5]) and views on the subject obtained from surveys conducted among the E&V Team and appropriate ARPANET-MILNet Interest Groups.

4.7 STARS-SEE OPERATIONAL CONCEPT DOCUMENT

Citations:

[STARS-SEE 1985] "Proposed Version 001.0," STARS Joint Service Team for Software Engineering Environments, Stars Joint Program Office, October 1985.

Synopsis:

The Software Technology for Adaptable, Reliable Systems — Software Engineering Environment (STARS-SEE) Operational Concept Document (OCD) presents requirements from the perspective of the STARS-SEE users. It represents a consensus among the Government agencies responsible for SEE development and support, STARS-SEE implementors, and potential users. Major sections of the document describe the STARS-SEE mission, operational and support environments, and system components and functions. The primary mission centers on the development, support, reuse, management, and control of mission critical software. The STARS-SEE system is defined to consist of the people, computers, software, and procedures needed to perform the mission. Major topics discussed include (1) the types of users and associated software activities, (2) the function of the Integration and Compatibility Framework, (3) the capabilities required by the Information Storage and Retrieval System, (4) the functional capabilities of the SEE, (5) the SEE-user interaction, and (6) the hardware and software characteristics of the computer system. The functional capabilities address project planning and control, requirements specification and analysis, design specification and analysis, software prototyping and modeling, reusability, program generation and unit testing, integration testing, quality assurance, verification and validation, configuration management, software/hardware integration, post deployment software support, project communications, generation of documents, data collection, performance and productivity measurement, help and training for STARS-SEE users, the transition to and tailoring of the STARS-SEE, and knowledge engineering.

4.8 GRUND, ET AL.: KEY CHARACTERISTICS OF APSES

Citations:

[Grund 1985] E.C. Grund, L.A. Hilliard, and K.A. Younger, "Key Characteristics of Ada Programming Support Environments," MITRE Corporation, ESD — TR-85-144, MTR-9590, July 1985, DTIC Number AD B096 137.

Synopsis:

This document is intended to provide basic information about Ada Programming Support Environments for people concerned with the specification or selection of an APSE. Section 1 summarizes the STONEMAN APSE requirements. Section 2 describes desirable characteristics of APSEs in five areas: compilers, run-time environments, databases, configuration management tools, and editors. A short list of questions to ask in each area is included. Section 3 describes four Ada programming support products available or under development in early 1985 in terms of their capabilities in the same five areas.

4.9 Ada-EUROPE: SELECTING AN Ada ENVIRONMENT

Citations:

[Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986.

Synopsis:

The Ada-Europe Environment Working Group, under the chairmanship of John Nissen, produced a guide which adopts the "point of view of a potential user wishing to select an environment, and provides lists of questions to be asked about the environment under consideration." It generally follows the structure proposed in Stoneman [@DoD 1980]; it "starts from the inside of the onion structure and works outwards." Each of its 19 chapters follows a standard format. Topics are introduced and discussed, typically using one or two pages of text, and then a list of appropriate questions is provided.

The structure is represented by the table of contents of the guide, reproduced in part below.

Part A Host and Target Considerations
2. Underlying machine
3. Target machine

Part B Kernel
4. Database, schema and typing
5. Versions, configurations and history
6. Security and integrity
7. Language issues and run-time support
8. Interaction between programs

Part C Aids for Tool Building
9. Meta-tools and tool components

Part D Man-Machine Interaction
10. Administrative aspects
11. The user interface
12. Help, error and warning messages

Part E Tool Functions
13. Office automation aspects
14. Static analysis, compilation and the program library
15. Testing, debugging and dynamic analysis
16. Project and product management
17. Life cycle support

Part F Other Issues
18. Performance of the environment
19. Contractual matters

Methods:

[Ada-Europe Ada Environment Questionnaires 13.2]

4.10 MCDERMID AND RIPKEN: LIFE CYCLE SUPPORT IN THE Ada ENVIRONMENT

Citations:

[McDermid 1984] J. McDermid and K. Ripken, "Life Cycle Support in the Ada Environment," Cambridge University Press, 1984.

Synopsis:

This book contrasts its own approach to APSE development with that of the Stoneman report [@DoD 1980]. Stoneman takes a bottom-up approach, starting with a kernel and minimal APSE (KAPSE and MAPSE), as a foundation for extensions to more powerful and better integrated environments. McDermid and Ripken follow a top-down approach by defining requirements for a coherent life-cycle methodology. They then describe a particular instance of a coherent methodology, as a combination of existing methods used in various life-cycle phases. This description becomes the basis for a definition of a "coherent APSE" that supports the entire life cycle.

The authors use a seven-phase life cycle and state requirements for each phase in terms of (1) a system *representation* form, (2) a *transformation* method and (3) a *verification* activity. Table 4.10-1 lists the names of the seven phases (each named for its principal output) and the methods selected for each.

Table 4.10-1 Example Coherent Methodology

PHASE (OUTPUT)	SELECTED METHOD
Requirements Expression	CORE
System Specification	A-7 Techniques
Abstract Functional Specification	A-7 Techniques
Module Specification	Ada and ANNA
Module Design	Ada and ANNA
Module Code	Ada and ANNA
Executable System	--

The authors are not completely satisfied with all of the methods chosen, and point out shortcomings in each case. They suggest the book be used as "a reference point for further work on APSE design and development." They stress that the coherence of the methods and ease of transition from one phase to the next is an important attribute. They also outline a phased development plan in which a larger scale APSE might be developed in the following three steps: (1) a "Clerical Support APSE," (2) a "V&V and Management Support APSE," and (3) a "Transformation Support APSE."

4.11 NOTKIN AND HABERMANN: SOFTWARE DEVELOPMENT ENVIRONMENT ISSUES AS RELATED TO Ada

Citations:

[Notkin 1981] D.S. Notkin and A.N. Habermann, "Software Development Environment Issues as Related to Ada," in "Tutorial: Software Development Environments," ed. A.I. Wasserman, IEEE, 1981, pp. 107-133.

Synopsis:

This paper addresses software development problems that arise in three areas: programming, system composition, and management. In each area traditional methods and tools are contrasted with a more integrated approach exemplified by an experimental environment named Gandalf.

"Programming issues are those that arise when a single programmer takes a program all the way from its specifications to a working program."

"System composition issues are those that arise when a system (or a version of a system) is built by integrating many programs into one." "The two basic problems in system composition are interface control and version control." Traditional methods use isolated tools "coordinated by memory...or scraps of paper."

"Management issues are those that arise when a group of more than one person develops and maintains a system over a period of time." Three problem categories are addressed: misunderstanding, lack of information, and conflict of interest. Traditionally, these problems have been handled by non-technical means. The problem with the management approach to a management environment is that the solution to human interaction difficulties is treated by the introduction of more human interaction.

Methods:

Although this paper was not written as an example of E&V technology, the following list of environment software requirements (paraphrased from the paper) may be used as a high-level checklist:

- Concurrent multiple users must be supported
- An efficient implementation of Ada must be possible
- Efficient support for data base manipulations is needed
- A good file system is essential
- An extensible command language is needed.

It is also pointed out that the most important hardware requirement is that the software requirements listed above must be supported.

4.12 STENNING, ET AL.: THE Ada ENVIRONMENT: A PERSPECTIVE

Citations:

[Stenning 1981] V. Stenning, T. Froggart, R. Gilbert, and E. Thomas, "The Ada Environment: a Perspective," in "Tutorial: Software Development Environments," ed. A.I. Wasserman, IEEE, 1981, pp. 36-46.

Synopsis:

This paper discusses the objectives and the design of the Ada Programming Support Environment. It is strongly influenced by the United Kingdom Ministry of Defense Ada Support System Study, which was initiated by the MoD in January 1979. According to the paper, the DoD KAPSE/MAPSE/APSE approach is strongly recommended to achieve portability. The APSE should be designed to support a project throughout its life cycle. Furthermore, it should be an open-ended environment. This would allow for the user to extend or modify existing tools. A basic configuration control manager, a complete user interface, and a complete basic tool set are necessary to develop an Ada Environment which will improve program reliability, life-cycle program costs, and promote portability.

4.13 WEIDERMAN: EVALUATION OF Ada ENVIRONMENTS

Citations:

[Weiderman 1987] N. Weiderman and N. Haberman, "Evaluation of Ada Environments," Software Engineering Institute, Technical Report CMU/SEI-87-TR-1, March 1987, DTIC Number AD A180 905.

Synopsis:

In response to the lack of available research about the selection of APSEs, the Software Engineering Institute (SEI) has developed a methodology to evaluate these environments. The methodology centers around the execution of several experiments in the environment to be evaluated. Several experiments have been developed in the following areas: System Management; Configuration Management/Version Control; Design and Code Development; Unit Testing and Debugging. The environments are evaluated in terms of functionality, performance, user interfaces, and system interfaces. The need for an evaluator to tailor an evaluation technique to a specific environment is addressed by the SEI study. The experiments that have been designed are generic experiments. The evaluator derives, or "instantiates," the environment-specific technique from the generic experiment. In the final phase of the evaluation, the results are analyzed. An advantage of the application of this methodology is that results can be compared from one environment to another. See also a paper describing an application of the SEI's method [@Gray 1987].

Methods:

[SEI Unit Testing and Debugging Experiment 7.2;
SEI Design Support Experiment 9.1;
SEI Configuration Management Experiment 10.2]

4.14 BARSTOW AND SHROBE: OBSERVATIONS ON INTERACTIVE PROGRAMMING ENVIRONMENTS

Citations:

[Barstow 1981] D.R. Barstow and H.E. Shrobe, "Observations on Interactive Programming Environments," in "Tutorial: Software Development Environments," ed. A.I. Wasserman, IEEE, 1981, pp. 285-301.

Synopsis:

This paper reviews key features of LISP-based environments and comments upon lessons learned from these environments and future directions. These environments encourage a "progressive enrichment" style of development rather than developments broken into distinct phases such as specification, implementation, and maintenance. The following set of lessons (described more fully in the paper) are concerned with the programmer's perception of the environment:

- It is important to be able to run an incomplete program.
- The user should be able to view the program from many different natural viewpoints, most of which are "structured" in nature.
- Intercommunication among tools is extremely important.
- The programmer should not be required to know the details of the particular language definition used in the current implementation.
- The environment's interface must be highly tuned to be as natural as possible for the human programmer.

Environment characteristics created with these lessons in mind "lead to the ultimate goal of a programming environment (which is to increase the ability of the programmer to communicate with the computer) by taking advantage of as many naturally occurring structures as possible."

4.15 HOUGHTON AND WALLACE: CHARACTERISTICS AND FUNCTIONS OF SOFTWARE ENGINEERING ENVIRONMENTS: AN OVERVIEW

Citations:

[Houghton 1987] R.C. Houghton, Jr. and D.R. Wallace. "Characteristics and Functions of Software Engineering Environments: An Overview," ACM Software Engineering Notes, Vol. 12 Number 1, January 1987.

Synopsis:

This paper provides a comprehensive discussion of software engineering environments in general, with no focus on Ada or any specific language. Some major topics discussed are:

- Environment Types and Life Cycle Relationships
- Integration
- Human Factors
- Analysis and Software Quality
- Support for Different Types of Users
- Support for Application
- Hardware Support
- Levels of Support.

In its concluding section, the paper stresses that software engineering environments should be viewed as systems that support broad categories of users and tasks throughout the full life cycle.

4.16 CAIS: DoD-STD-1838

Citations:

[DoD 1986] DoD-STD-1838, Common APSE Interface Set (CAIS), U.S. Department of Defense, 9 October 1986, DTIC Number AD A157 589.

Synopsis:

DoD-STD-1838, hereafter called CAIS, was developed by the KAPSE Interface Team (KIT) and the KAPSE Interface Team for Industry and Academia (KITIA) during the period from 1981 to 1986 as a first evolutionary step towards a full state-of-the-art common APSE interface standard.

The CAIS is designed to promote source-level portability of Ada programs, especially Ada software development tools. The goal of the CAIS is to promote interoperability (of database objects) and transportability (of APSE tools) of Ada software across Department of Defense (DoD) APSEs. See also [@MIL 1989] and synopsis [4.17], and the overview paper [@Oberndorf 1988].

4.17 CAIS-A: MIL-STD-1838A

Citations:

[MIL 1989] "Common APSE Interface Set, Revision A," MIL-STD-1838A, April 1989, DTIC Number AD A157 589.

Synopsis:

CAIS-A is a set of Ada package interfaces designed to enhance the transportability of Ada Support Environment Tools. The scope of the CAIS-A includes the functionality affecting transportability that is needed by tools, but not provided by the language. The CAIS-A contains definitions for an entity management system for software engineering tools. The primitive entities defined allow for the manipulation of devices, files, and processes. CAIS-A is based on an entity-relationship approach and it allows the user to define entities, in a limited way, by means of a typing mechanism. CAIS-A also includes functionality to support tools requiring transaction processing, a rudimentary triggering mechanism, and explicit control over APSE distribution.

The CAIS-A was developed by SofTech under contract to Naval Ocean Systems Center. CAIS-A is a design enhancement to the existing CAIS (DoD-STD-1838) [@DoD 1986]; see synopsis [4.16], which was developed by the KIT and KITIA as a first evolutionary step towards a full, state-of-the-art interface standard. Designers view CAIS-A as the next step in that evolutionary process.

4.18 HOGAN AND PRUD'HOMME: DEFINITION OF A PRODUCTION QUALITY COMPILER

Citations:

[Hogan 1985] M.O. Hogan, and S.M. Prud'homme, "Definition of a Production Quality Compiler," Aerospace Corporation, Technical Report, July 1985, DTIC Number AD A182 445.

Synopsis:

The study that led to this report was sponsored by the Space Division of the Air Force Systems Command. The report "is organized as a set of prototype requirements, along with guidance on how to tailor the requirement for specific application areas. In this form it can be used either as a tool to help determine whether a particular compiler is of production quality or as a guide for preparing requirements for compilers to be used in the development and maintenance of software for mission critical computer resources."

Chapters 2 through 6 address interface requirements: user interfaces, machine interfaces, runtime system interfaces, compiler related components interfaces, and Ada language interfaces, respectively. Chapter 7 addresses capacity and performance requirements, Chapter 8 addresses reliability requirements, Chapter 9 addresses documentation requirements. Each of the above chapters follows a standard format in which a requirement is stated in the form: "The compiler shall . . . ," and then a "Guidance" section is provided giving background information and help in subsetting or tailoring the requirement for specific application domains.

4.19 NISSEN, ET AL: GUIDELINES FOR Ada COMPILER SPECIFICATION AND SELECTION

Citations:

[Nissen 1984] J.C.D. Nissen, B.A. Wichman, et al., "Guidelines for Ada Compiler Specification and Selection," in Ada: *Language, Compilers and Bibliography*, ed. M.W.Rogers, Cambridge University Press, 1984.

Synopsis:

Members of Ada-Europe produced this set of guidelines based upon a taxonomy of compiler features. Their caveat is clear: "The relative value of information about different features of the compiler is a matter of judgment and circumstance. ... It is the reader's responsibility to weigh each factor according to his requirements. No liability of whatever kind shall be carried by the authors."

The taxonomy is represented by the table of contents of the guide, reproduced in part below.

2. Host and target
3. Language-related issues
4. User-interfacing and facilities
   - 4.1 Compiler invocation and listing management
   - 4.2 Compilation options
   - 4.3 Other features
   - 4.4 Errors and warnings
   - 4.5 Other software supplied
   - 4.6 Compilation management
5. Performance and capacity
   - 5.1 Host performance and capacity
   - 5.2 Target code performance
6. Compiler and run-time interfacing
   - 6.1 Compiler issues
   - 6.2 Run-time system issues
7. Retargetting and rehosting
   - 7.1 Introduction and definitions
   - 7.2 Retargetting
   - 7.3 Rehosting
8. Contractual matters
9. Validation

Chapter 2, Host and Target, briefly treats compiler configuration issues, and provides a questionnaire [Host and target questionnaire 14.1].

Chapter 3, Language-related issues, extracts from the Ada language reference manual [@DoD 1983] those features explicitly allowed to vary based upon machine specific characteristics [Machine-specific characteristics questionnaire 14.2].

Methods:

[Host and target questionnaire 14.1;
Machine-specific characteristics questionnaire 14.2]

4.20 WIS COMPILER EVALUATION GUIDELINES

Citations:

[WIS CEG 1985] "WIS Compiler Evaluation Guidelines," GTE Labs, Technical Report, 1985.

Synopsis:

This document presents guidelines that provide an information base on which specific compiler evaluation methodology and criteria can be built. Three types of guidelines have been identified: essential characteristics, highly recommended characteristics, and recommended characteristics. Also, certain questions that compiler vendors should be asked regarding their compilers measurable characteristics are listed. The guidelines take the view that the development of Ada compilers is an ongoing process. To address this fact, the document discusses, where appropriate, general aspects of compilers, and specific aspects of Ada compilers.

The document is broken down into four main sections. Part 1 is an introductory section. Part 2 provides background information on Ada compilers. Part 3 discusses compiler architecture issues. Part 4 then provides the main Ada compiler guidelines.

4.21 WIS TOOL EVALUATION CRITERIA

Citations:

[WIS CEC 1985] G. Gicca and C. Stacey, "Component Evaluation Criteria," GTE Government Systems, Technical Report, 16 August 1985.

Synopsis:

This document outlines a process to be used in evaluating currently available software tools for inclusion in an Ada software development environment. It defines a four phase evaluation process where each successive phase takes a more detailed view of the particular development tool. All phases have the same basic set of evaluation categories, with the definition being refined in the following phase. There are seven such categories. These are: "Functional Applicability," "Understandability," "Testability," "Evolvability," "Efficiency," "Portability," and "Human Engineering." Their definitions at a high level are:

Functional Applicability — the extent to which the tool or component fulfills a current need within a software development support environment

Understandability — the extent to which the tool or component is understandable from a systems viewpoint

Testability — the ease with which a program can be tested to verify that it performs its intended functions

Evolvability — a category that evaluates the combination of both modifiability and expandability

Efficiency — the amount of time and space required by a program to perform a function

Portability — the ease of transferring a program from one hardware configuration or software environment to another

Human Engineering — the ease of learning, operating, preparing input, and interpreting output of a program.

Each of the four phases of the component evaluation has its own rating scheme. A checklist is created for each category and for each sub-category within the basic evaluation categories. The rating scheme itself defines a set of numbers from 1 to 5. The reviewer then rates a particular tool or component by assigning a value for each sub-category. A weighting factor is then used to prioritize sub-categories and main categories. In the end a final set of numbers is produced that allows for overall comparisons between tools that offer similar capabilities.

4.22 WEIDERMAN: COMPILER EVALUATION AND SELECTION

Citations:

[Weiderman 1989] N.H. Weiderman, "Ada Adoption Handbook: Compiler Evaluation and Selection, Version 1.0," Software Engineering Institute, CMU/SEI-89-TR-13, March 1989, DTIC Number AD A207 717.

Synopsis:

The following Abstract is quoted directly from the cited document:

"The evaluation and selection of an Ada compilation system for a project is a complex and costly process. Failure to thoroughly evaluate an Ada compilation system for a particular user application will increase project risk and may result in cost and schedule overruns. The purpose of this handbook is to convince the reader of the difficulty and importance of evaluating an Ada compilation system (even when there is no freedom of choice). The handbook describes the dimensions along which a compilation system should be evaluated, enumerates some of the criteria that should be considered along each dimension, and provides guidance with respect to a strategy for evaluation. The handbook does not provide a cookbook for evaluation and selection. Nor does it provide information on specific compilation systems or compare different compilation systems. Rather it serves as a reference document to inform users of the options available when evaluating and selecting an Ada compilation system."

The chapter headings are as follows:

1. Introduction
2. Common Questions
3. Compiler Validation and Evaluation
4. Practical Issues of Selecting an Ada Compiler
5. Compile/Link-Time Issues
6. Execution-Time Issues
7. Support Tool Issues
8. Benchmarking Issues
9. Test Suites and Other Available Technology

In Chapters 4 through 8 there are a number of lists (some annotated) of criteria, factors, features, and questions to be asked. Some of these are synopsized in, or have influenced, later sections of this document, as indicated below.

Methods:

[Weiderman: Compiler Evaluation Lists 5.13;
Vendor Evaluation Questionnaire 99.7]

5. COMPILATION SYSTEM ASSESSORS

For the purposes of this document, the compilation system is defined as those APSE components which are Ada-specific and are required for validation: the compiler, the code generator, the program library management system, and the runtime support system. While each of these components have characteristics which should be assessed individually, the assessment of their combined functionality will be more critical to the successful development of mission critical software.

The criticality of assessor development for these four compilation system components is made evident by the many large-scale projects with requirements for the use of Ada. These large-scale projects include the Strategic Defense Initiative (SDI), the NASA Space Station, the Software Technology for Adaptable, Reliable Systems (STARS) program, Army Tactical Command and Control System, Army WWMCCS Information System (WIS), and the Advanced Tactical Fighter (ATF), A-12, and Light Helicopter Experimental (LHX) programs being evaluated for common avionics systems under the auspices of the Joint Integrated Avionics Working Group (JIAWG). The successful performance of these systems depends upon the quality/extent of code generation support and execution support found in the compilation system.

5.1 Ada COMPILER VALIDATION CAPABILITY (ACVC)

Purpose: Validation of the completeness of the Ada compiler by means of a compiler test suite. The ACVC consists of a test suite, analysis tools, and accompanying documentation, to enable the determination of conformance of Ada compiler implementations to the ANSI/MIL-STD-1815A. Note: The AJPO requires that Ada compilers pass the ACVC and the vendor allow the distribution of the resulting Validation Summary Report (VSR) in order for the compiler to be advertised as a commercially available Ada compiler. [@RM: Compilation 7.1.6.7, @RM: Completeness 6.4.9]

Primary References:
[ACVC 1989] Ada Compiler Validation Procedures, Version 2.0, AJPO, May 1989.

Vendors/Agents: [National Technical Information Service]

Method: Automated test suite.

Inputs:

ACVC source code, Ada compiler and runtime system, and host (and target) computer.

Process:

1. Obtain latest ACVC test suite
2. Following documentation, compile and run tests
3. Use analysis tools on test outputs.

Outputs:

Validation results;
Validation Summary Report (VSR).

5.2 IDA BENCHMARKS

Purpose: Evaluation of the capacity and performance of the Ada compiler by means of a compiler test suite.
[@RM: Compilation 7.1.6.7, @RM: Capacity 6.4.6;
@RM: Processing Effectiveness 6.4.22; @RM: Storage Effectiveness 6.4.31]

Primary References:
[IDA 1985] A.A. Hook, G.A. Riccardi, M. Vilot, and S. Welke, "User's Manual for the Prototype Ada Compiler Evaluation Capability (ACEC)," Version 1, Institute for Defense Analysis, IDA Paper P-1879, October 1985, DTIC Number AD A163 272.

Host/OS: VAX/VMS or any system that can read ANSI standard tapes.

Vendors/Agents: [SofTech, Inc.]

Method: Test suite.
Inputs: IDA source code, Ada compiler and runtime system, and host (and target) computers.
Process:
1. Obtain test suite from agent
2. Compile and run Ada programs.

Outputs: Timing and storage measurements for individual language features.

5.3 Ada COMPILER EVALUATION CAPABILITY (ACEC)

Purpose: The purpose of this test suite is best stated by the following quote taken from the introduction in the ACEC Reader's Guide: "The ACEC .... consists of a portable test suite and support tools. ... It contains test problems designed to measure the execution time and size of a systematically constructed set of Ada examples. The support tools assist the ACEC user in executing the test suite and analyzing the results obtained." The scope of coverage provided by the test suite is shown by the following excerpts from the ACEC classification taxonomy:

II. Execution Time Efficiency
  A. Language Feature Efficiency
    1. Required (referenced by LRM section)
    2. Implementation Dependent (referenced by LRM section)
      - attributes (LRM Appendix A)
      - record representation clauses
      - interrupts
      - language interface
      - unchecked programming
  B. Pragmas
    1. Predefined
    2. Implementation Defined
  C. Optimizations
    1. Classical
    2. Effects of Pragmas
    3. Static Elaboration
      - aggregates
      - tasks
    4. Language Specific
      - Habermann-Nassi transformation for tasking
      - delay statement optimization
  D. Performance Under Load
    1. Task Loading
      - task creation
      - task termination
      - task abortion
      - Dining Philosophers Problem
      - task starvation
      - task delay
    2. Levels of Nesting
      - static
      - dynamic
    3. Parameter Variation
    4. Declarations

E. Trade Offs
   1. Design Issues
      - order of evaluation
      - default vs initialized records
      - order of selection (rendezvous)
      - scope of usage (global, local, shared)
   2. Coding Style Variation

F. Operating System Kernel Efficiency
   1. Task Scheduling
   2. Exception Handling
   3. File I/O
   4. Memory Management/Storage Reclamation
   5. Elaboration
   6. Run Time Checks
   7. Interrupt Handling

G. Application Profile Tests
   1. Classical
      - Whetstone
      - Dhrystone
      - Ackerman's
      - Computer Family Architecture
      - Sort Variations
      - Math Applications
      - Livermore Loops
      - Knuth Loops
   2. Ada in Practice
      - E-3A simulator
      - navigation algorithms
      - radar tracking algorithms
      - communication algorithms
      - electronic warfare
      - avionics
   3. Ideal Ada
      - AI applications
      - data encryption

III. Code Size Efficiency
   A. Expansion Code Size
   B. Run Time System Size
   C. Executable File Size

The first version of the ACEC was released in August 1988.
[@RM: Compilation 7.1.6.7, @RM: Processing Effectiveness 6.4.22;
@RM: Storage Effectiveness 6.4.31]

Primary References:
[ACEC 1988] "Ada Compiler Evaluation Capability (ACEC) Technical Operating

Report (TOR) Reader's Guide," Air Force Wright Aeronautical Laboratory, Document Number D500-11790-2, 10 August 1988, DTIC Number AD B125 147.

Vendors/Agents: [DACS]

Method: Automated test suite

Inputs: ACEC source code, Ada compiler and runtime system, host and target computer.

Process:

1. Obtain the ACEC
2. Compile and run the tests according to the documentation
3. Use the analysis tools on the test outputs.

Outputs: Reports containing the execution time and/or code size for the selected tests.

5.4 PIWG BENCHMARK TESTS

Purpose: Identification of performance characteristics of Ada compilers. A set of tests have been collected by the Performance Issues Working Group (PIWG) of the Special Interest Group for Ada (SIGAda) of the Association for Computing Machinery (ACM). The tests examine the performance of the compiler itself in terms of compilation speed, as well as the quality of the generated code for both processing and storage effectiveness. The test suite measures performance for both isolated language features and composites or mixes of language features (using the Whetstone and Dhrystone tests).

[@RM: Compilation 7.1.6.7, @RM: Processing Effectiveness 6.4.22; @RM: Storage Effectiveness 6.4.31]

Primary References:

Vendors/Agents: [PIWG]

Method: Automated test suite.

Inputs: PIWG source code, Ada compiler and runtime system, and host (and target) computer.

Process:

1. Obtain the latest PIWG tests
2. Compile and run tests according to the documentation.

Outputs: Reports on the outcome of each test run.

5.5 UNIVERSITY OF MICHIGAN BENCHMARK TESTS

Purpose: Identification of the execution efficiency of the code generated by Ada compilers. The tests measure only the performance of isolated language features as they relate to real-time performance.
[@RM: Compilation 7.1.6.7, @RM: Processing Effectiveness 6.4.22]

Primary References:
[Mich 1986] R.M. Clapp, L. Duchesneau, R.A. Volz, T.N. Mudge, and T. Schultze, "Toward Real-Time Performance Benchmarks for Ada," Electrical Engineering and Computer Science Dept., Univ. of Michigan, RSD-TR-6-86, January 1986, pp. 1-25.

Vendors/Agents: [UMich]

Method: Test suite.
Inputs: UMichigan source code, Ada compiler and runtime system, and host (and target) computer.
Process:
1. Obtain the UMichigan tests
2. Compile and run according to the documentation.

Outputs: Reports on the outcome of each test run.

5.6 MITRE BENCHMARK GENERATOR TOOL (BGT)

Purpose: Evaluation of the ability of an Ada compilation system to support development of very large systems in Ada. Under sponsorship of the Federal Aviation Administration, MITRE developed the Benchmark Generator Tool (BGT). The benchmark tests address capacity issues arising with large system developments. The initial version (1988) includes two types of tests: Library Capacity Tests and Dependency Maintenance Tests.
[@RM: Compilation 7.1.6.7, @RM: Capacity 6.4.6]

Primary References:
[MITRE BGT 1986] S.R. Rainer, and T.P. Reagan, "User's Manual for the Ada Compilation Benchmark Generator Tool," MITRE Corporation, MTR-87W00192-01, January 1988.

Host/OS: Any for which an Ada compiler exists.

Vendors/Agents: [MITRE, McLean, VA]

Method: Automated tool.
Inputs: BGT source code, Ada compiler, and host computer.
Process:
1. Obtain the BGT
2. Compile according to the documentation.

Outputs: Results of the above analysis, including capacity limits, link time, compilation time, etc.

5.7 UK Ada EVALUATION SYSTEM (AES)

Purpose: Evaluation of Ada compilers and associated linkers/loaders, program library systems, debuggers, and run-time libraries. A test suite and a methodology (AES) were developed by Software Sciences Ltd., under sponsorship of the UK Ministry of Defense (MoD). The British Standards Institute (BSI) has been sponsored by the MoD to provide an Ada Evaluation Service, using the AES. Interested parties, such as compiler vendors or potential compiler purchasers, may pay BSI to conduct an evaluation or to supply a copy of an existing evaluation report.

Primary References:

[UK AES 1986] R.H. Pierce, I. Marshall, and S.D. Blude, "An Introduction to the MoD Ada Evaluation System," Software Sciences Ltd., Report Number 5485, June 1986.

Host/OS: Any for which an Ada compiler exists.

Vendors/Agents: [BSI, Milton-Keynes, UK]

Method: Automated test suite and questionnaire.

Inputs: AES source code and questionnaire, Ada compiler and runtime system, and host (and target) computer.

Process: Pay BSI to do an evaluation or purchase an existing evaluation report.

Outputs: An Evaluation Report organized in a standard format.

5.8 COMPILATION CHECKLIST

Purpose: Evaluation of the power of compilation by developing a list of functional capabilities.
[@RM: Compilation 7.1.6.7, @RM: Power 6.4.21]

Primary References:
[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.
[Classification Schema/E&V Taxonomy Checklists: 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capability checklist (see Table 5.8-1) and compiler documentation.

Process: Check off capabilities demonstrated during compiler runs or discussed in the documentation.

Outputs: A list of capabilities provided by the compiler.

Table 5.8-1 Compilation Capabilities Checklist

FEATURE	FOUND
Conditional Compilation	
Incremental Compilation	
Debug Information Generation	
Enable/Disable Listing	
Errors Only Listing	
Error Identification	
Set Default Directory For Source	
Set Listing Width And Height	
Specify Different Program Library	
Specify Main Program	
Disable Use Of SYSTEM Library	
Change package SYSTEM	
Suppress All Run-Time Checks	
Compile Multiple Files	
Language Sensitive Editor Support	
Specify Error Limit	
Enable/Disable An Error Category	
Specify Optimization Parameters	
Syntax Only Checking	
Symbol Table	
Variable Set/Use Indications (Cross Reference)	
Object Code [illegible]ng	
Object Attribute Map	
Code Statistics	
Unidentified Compiler Options(Pragmas)	
Controlled Dynamic Storage	
Elaboration Control	
Inline Expansion Of Subprograms	
Interface With Other Languages	
Specify Memory Size	
Pack Data Representations In Memory	
Priority Control Of Concurrent Tasks	
Shared Variables	
Specify Storage Unit	
Specify Alternative System Characteristics	
Machine Code Mapping	
Machine Code Insertions	
Cross Compilation	
Error Reporting	
Exceptions List	
Identify Target Dependencies	
Save User Configuration	
Shared Generic Support	

5.9 PROGRAM LIBRARY MANAGEMENT CHECKLIST

Purpose: Evaluation of the completeness and power of program library management by developing a list of functional capabilities.
[@RM: Program Library Management 7.2.1.7, @RM: Completeness 6.4.9; @RM: Power 6.4.21]

Primary References:
[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.
[Classification Schema/E&V Taxonomy Checklists: 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 5.9-1) and program library manager documentation.

Process: Check off capabilities demonstrated by the program library manager or discussed in the documentation.

Outputs: A list of capabilities provided by the program library manager.

Table 5.9-1 Program Library Management Capabilities Checklist

FEATURE	FOUND
Listing Information	
List Of Logical Unit Names	
Associated File Names For Unit	
Units Using Specified Unit	
Units Used By Specified Unit	
Size Information	
Time-Stamp Information	
Kind And Granularity Of Compilation Element	
Units Used To Construct Executable	
Completeness And Currency Check	
Automatic Build Capability	
Automatic Compilation/Recompilation	
Spawn Command Language Subprocess	
Create Structures	
Move Elements Between Libraries	
Move Elements Between Directories	
Remove Compilation Unit	
Obsolescence Management	
Library Access Control	
Read Only (Shared)	
Exclusive	

5.10 ARTEWG CATALOGUE OF Ada RUNTIME IMPLEMENTATION DEPENDENCIES

Purpose: The purpose of this document is best stated by the following quotation taken from the rationale section in the catalogue: "The main goal of this catalogue is to be the one place where all the areas of the Ada Reference Manual ... which permit implementation flexibilities can be found." These implementation dependencies affect the performance and adaptation characteristics of the generated code. The text describes each known dependency by a number (which identifies the relevant section and paragraph in the Ada Reference Manual), a topic or title, a question which poses the implementation issue, a dependency type (either explicit or implicit), a rationale explaining why the dependency exists, and an Ada example to further clarify the dependency. (These descriptions could be used as the basis for an automated test suite.)

[@RM: Compilation 7.1.6.7, @RM: Anomaly Management 6.4.2;
@RM: Processing Effectiveness 6.4.22; @RM: Retargetability 6.4.27;
@RM: Storage Effectiveness 6.4.31]

Primary References:

[ARTEWG 1987] Catalogue of Ada Runtime Implementation Dependencies," Association for Computing Machinery, Special Interest Group on Ada, Ada Runtime Environment Working Group, 1 December 1987.

Vendors/Agents: [ARTEWG]

Method: Questionnaire.

Inputs: Descriptions of implementation dependent features.

Process:

1. Select critical dependencies
2. Build and run tests for each dependency or ask vendor how dependencies are implemented
3. Select compiler and/or make coding standards based on results of step 2.

Outputs: Evidence showing how features are implemented.

5.11 ARTEWG RUNTIME ENVIRONMENT TAXONOMY

Purpose: Describes the basic elements of Ada runtime environments and provides a common vocabulary. The following excerpt is taken from the introduction to the Taxonomy section. "If a runtime environment for an Ada program is composed of a set of data structures, a set of conventions for the executable code, and a collection of predefined routines, then the question arises: what are examples of these elements, and moreover, what is the complete set from which such elements are taken when a particular runtime environment is built?...It should be noted that the dividing line between the predefined runtime support library on one hand, and the conventions and data structures of a compiler on the other hand, is not always obvious. One Ada implementation may use a predefined routine to implement a particular language feature, while another implementation may realize the same feature through conventions for the executable code. ... This taxonomy concerns itself primarily with those aspects of the runtime execution architecture which are embodied as routines in the runtime library. It does not treat issues of code and data conventions, nor issues related to particular hardware functionalities, in any great depth."
[@RM: Runtime Environment 7.2.3.5, @RM: Completeness 6.4.9]

Primary References:
[ARTEWG 1988] "A Framework for Describing Ada Runtime Environments," Proposed by Ada Runtime Environment Working Group (SIGAda), Ada Letters, Volume VIII, Number 3, May/June 1988, pp. 51-68.

Vendors/Agents: [ARTEWG]

Method: Questionnaire

Inputs: Questionnaire (see Table 5.11-1) and runtime environment documentation.

Process: Describe capabilities demonstrated by the runtime environment or discussed in the documentation.

Outputs: A description of capabilities provided by the runtime environment.

Table 5.11-1 Runtime Environment Taxonomy

CAPABILITY	DESCRIPTION
Runtime Execution Model Generated Code Number of Areas for Package Data Mechanism for Uplevel Referencing of Objects (static link or display) Subprogram Call Sequences Parameter Passing Mechanisms Register Usage Preservation of Registers across Subprogram Calls Representation of Pointers Implementation of Runtime Type Checks Data Structures in the RTE Division between Inline Code and Runtime Routines Tasking Constructs Memory Management Exception Management Attributes Miscellaneous Commonly Invoked Routines Use of Target Instruction Set Architecture User-Visible Interfaces to Extend the Runtime System Dynamic Memory Management Stack Management Heap Management Single Heap or One Heap/Task Arrangement of Storage for Collections Storage Reclamation None Explicit (Unchecked Deallocation) Garbage Collection Pragma Controlled Processor Management Block Tasks Unblock Tasks Selection of Tasks which Actually Run Task Priorities Assignment to Physical Processor	

Table 5.11-1 Runtime Environment Taxonomy (Continued)

CAPABILITY	DESCRIPTION
Interrupt Management Asynchronous Events Timer Interrupts I/O Interrupts Hardware Failures Others Program Synchronous Events Arithmetic Overflow Arithmetic Underflow Divide by Zero Others Address Clauses for Task Entries Interrupt Initialization Interrupt Resetting Time Management Package Calendar Delay Statement Implementation Exception Management Finds Exception Handler for Exception Transfers Control to Exception Handler Rendezvous Management Task Activation Task Termination Task Completion Task Termination Task Abortion I/O Management Predefined Packages from Chapter 14 of ARM Additional I/O Facilities	

Table 5.11-1 Runtime Environment Taxonomy (Continued)

CAPABILITY	DESCRIPTION
Commonly Called Code Sequences Multi-Word Arithmetic Operations Block Moves String Operations Attribute Calculations Others Target Housekeeping Functions Starting the Execution Environment Determining the Hardware Environment Determining the Software Environment Processor Initialization Interrupt Initialization Other Terminating the Execution Environment	

5.12 COMPILER ASSESSMENT QUESTIONNAIRE

Purpose: The document presents a hierarchical breakdown of the compiler shown in Fig. 5.12-1. Requirements for each element in the hierarchy are listed for certain attributes. Each requirement is augmented by one or more questions which address the requirement.
[@RM: Compilation 7.1.6.7, @RM: Attributes 6.]

Primary References:
[E&V Report 1984] "Requirements for Evaluation and Validation of Ada Programming Support Environments, Version 1.0," 17 October 1984, Appendix B of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, November 1984, pp. B-45 – B-85, DTIC Number AD A153 609.
[Requirements for E&V: 4.5]

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire and compiler documentation.

Process: Answer questions based on documentation, using the compiler, or asking the vendor.

Outputs: Completed questionnaire.

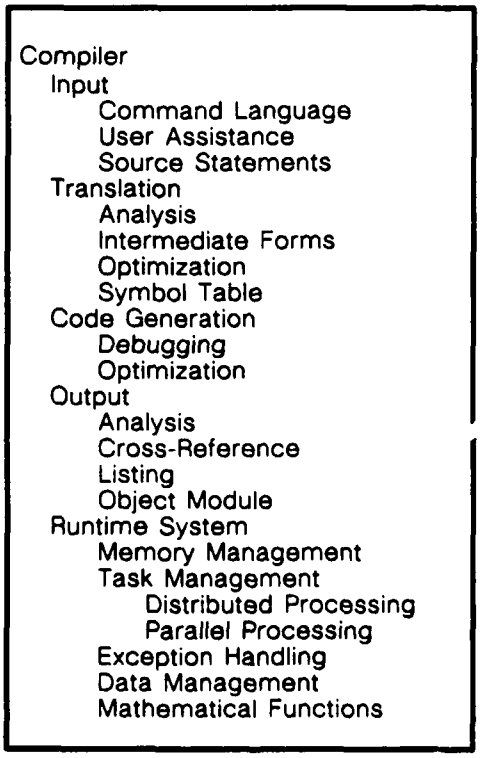

Figure 5.12-1 Compiler Hierarchy

5.13 WEIDERMAN: COMPILER EVALUATION LISTS

Purpose: This handbook "describes the dimensions along which a compilation system should be evaluated, enumerates some of the criteria that should be considered along each dimension, and provides guidance with respect to a strategy for evaluation." Table 5.13-1 below provides a "list of lists" found in Chapters 5, 6, and 8 of the handbook. Refer to the handbook itself for the actual elements of each list. In some cases the elements are simply listed; in other cases they are annotated with additional explanation and discussion.
[@RM: Compilation 7.1.6.7, @RM: Attributes 6.]

Primary References:
[Weiderman 1989] N.H. Weiderman, "Ada Adoption Handbook: Compiler Evaluation and Selection, Version 1.0," Software Engineering Institute, Technical Report CMU/SEI-89-TR-13, March 1989, DTIC Number AD A207 717.
[Weiderman: Compiler Evaluation and Selection: 4.22]

Vendors/Agents: [SEI]

Method: Checklists and Questionnaires

Inputs: Lists, characterization forms, and accompanying discussion (see Table 5.13-1).

Process: Employ lists to evalute appropriate dimensions, as indicated in the handbook.

Outputs: Lists of capabilities and completed characterization forms.

Table 5.13-1 Compiler Evaluation Lists

Compile/Link-Time Issues (Chapter 5)

- Compiler options often provided (5.1.1)
- Implementation-defined pragmas (5.1.2)
- Other important compiler features (5.1.4)
- Factors influencing "lines of code per minute" (5.2)
- Questions to be answered (5.2)
- Capacities and limits tested by the AES [5.7] (5.2)
- Documentation characteristics (5.4.3)

Execution-Time Issues (Chapter 6)

- Features *not* likely to generate calls to the runtime system (6.2.7)
- Features *likely* to generate calls to the runtime system (6.7.2)
- Optimizations that can be performed (6.2.3)
- Features critical to tasking performance (6.4.1)
- Operations concering exception handling whose overhead can be measured (6.4.2)
- Areas of concern related to space efficiency of the runtime system (6.5)
- Useful features of runtime systems (6.6)
- Questions related to interrupt handling (6.8)

Benchmark Issues (Chapter 8)

- Factors causing variation in results (8.2)
  - Memory effects
  - Processor effects
  - Operating and runtime system effects
  - Program translation effects
- Standard benchmark configuration information (8.7)
  - For the host system
  - For the target system
  - For all benchmarks

5.14 RUNTIME SUPPORT SYSTEM QUESTIONNAIRE

Purpose: Characterization and evaluation of the Runtime Support (RTS) system. [@RM: Runtime Environment 7.2.3.5; @RM: Anomaly Management 6.4.2; @RM: Communication Effectiveness 6.4.8; @RM: Completeness 6.4.9; @RM: Functional Scope 6.4.15; @RM: Generality 6.4.16; @RM: Granularity 6.4.17; @RM: Modularity 6.4.19; @RM: Retargetability 6.4.27; @RM: System Accessibility 6.4.32; @RM: System Compatibility 6.4.34]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire (see Fig. 5.14-1) and runtime support system documentation.

Process: Answer questions based on documentation, using the runtime support system, or asking the vendor.

Outputs: Completed questionnaire.

Is support provided for
- single processor only?
- multiple homogenous processors?
- multiple heterogenous processors?
  * calls made to another process?
  * actual synchronization?

Is support provided for
- single programs?
- multiple programs?

Is support provided for
- tight coupling (characterized by shared/common memory)
- loose coupling (communicate but no shared/common memory)

Does the RTS
- require use of the operating systems?
- accept instructions from the operation system?
- replace the operating system?

Is the RTS
- modularly constructed?
- modifiable — (standard modifications or user-defined)?
- sub-settable?
- fault tolerant?
- secure?

What are the language features that are supported?
- How are they supported?

Figure 5.14-1 Runtime Support System Questionnaire

6. TARGET CODE GENERATION AIDS AND ANALYSIS TOOLSET ASSESSORS

These tools are used to assess host-target system cross-assemblers; host-based target linkers and loaders; host-based target system instruction-level simulators/emulators; host-based target-code symbolic debuggers; and host-based target system instrumentation interfaces which provide visibility into target processes during program execution. These assessments are also used in the case where the host computer is also the target computer.

6.1 ASSEMBLING CHECKLIST

Purpose: Evaluation of the power of assembling by developing a list of functional capabilities.
[@RM: Assembling 7.1.6.6, @RM: Power 6.4.21]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 6.1-1) and assembler documentation.

Process: Check off capabilities demonstrated during assembler runs or discussed in the documentation.

Outputs: A list of capabilities provided by the assembler.

Table 6.1-1 Assembling Capabilities Checklist

FEATURE	FOUND
Code Generation	
Macro Preprocessing	
Conditional Assembly	
Debug Information Generation	
Enable/Disable Listing	
Errors Only Listing	
Set Listing Width and Height	
Suppress All Run-Time Checks	
Assemble Multiple Files	
Specify Error Limit	
Enable/Disable An Error Category	
Syntax Only Checking	
Symbol Table	
Code Statistics	
Cross Assembly	

6.2 LINKING/LOADING CHECKLIST

Purpose: Evaluation of the power of linking/loading by developing a list of functional capabilities.
[@RM: Linking/Loading 7.1.6.13, @RM: Power 6.4.21]

Primary References:
[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.
[Classification Schema/E&V Taxonomy Checklists: 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 6.2-1) and linker/loader documentation.

Process: Check off capabilities demonstrated during linker/loader runs or discussed in the documentation.

Outputs: A list of capabilities provided by the linker/loader.

Table 6.2-1 Linking/Loading Capabilities Checklist

FEATURE	FOUND
Non-Specific Language Linking	
Deferred (After A Specific Time)	
Enable/Disable Link Map Generation	
Specify Full/Brief Link Map	
Generate A Link Command File	
Enable/Disable Symbol Cross-Reference	
Generate Debug Information	
Enable/Disable Execution File Creation	
Specify Batch/Nobatch Operation	
Specify Map File Name	
Specify Object File Name	
Specify Diagnostic Output File	
Enable/Disable System Library Search	
Enable/Disable Traceback Information	
Library Search Capabilities	
Extended Options Capabilities	
Sharable Image Support	
Specify Maximum Memory	
Specify Optimization Parameters	
Force Load	
Enable/Disable Library Trace	
Specify Main Program	
Non-Specific Language Main Program	
Overlays	
Link-Time Dead Code Elimination	

6.3 IMPORT/EXPORT CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of import/export by developing a list of functional capabilities.
[@RM: Import/Export 7.2.3.6, @RM: Completeness 6.4.9]

Primary References:
[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.
[Classification Schema/E&V Taxonomy Checklists: 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 6.3-1) and import/export documentation.

Process: Check off capabilities demonstrated by the import/export system or discussed in the documentation.

Outputs: A list of capabilities provided by the import/export system.

Table 6.3-1 Import/Export Capabilities Checklist

FEATURE	FOUND
Host to Target Object Downloading Target to Host Data Uploading	

Note: This table will be expanded in a future version of the Guidebook.

6.4 EMULATION CAPABILITIES CHECKLIST

Purpose: Evaluation of the power of emulation by developing a list of functional capabilities.
[@RM: Emulation 7.3.2.13, @RM: Power 6.4.21]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 6.4-1) and emulation system documentation.

Process: Check off capabilities demonstrated by the emulation system or discussed in the documentation.

Outputs: A list of capabilities provided by the emulation system.

Table 6.4-1 Emulation Capabilities Checklist

FEATURE	FOUND
Session security (lock-out unauthorized users)	
RS-232 interface to host (portable among hosts)	
Replaceable target pods (portable among targets)	
Support for simulating hardware devices	
Switching screen (user vs. debug displays)	
Read/write access to program library symbols	
Runtime controls of the state of the emulator	
Read/write access to target memory and I/O	
Full-speed execution with active breakpoints	
Full-speed execution while tracing	
Dynamic window for variables	
Multi-task tracing	
Exception tracing	

6.5 DEBUGGING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness and power of debugging by developing a list of functional capabilities.
[@RM: Debugging 7.3.2.5, @RM: Completeness 6.4.9; @RM: Power 6.4.21]

Primary References:
[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.
[Classification Schema/E&V Taxonomy Checklists: 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 6.5-1) and debugger documentation.

Process: Check off capabilities demonstrated by the debugger or discussed in the documentation.

Outputs: A list of capabilities provided by the debugger.

Table 6.5-1 Debugging Capabilities Checklist

FEATURE	FOUND
Instrumentation	
Statement	
Branch	
Block	
CSU	
CSC	
Machine Level Debugging	
Host-Based Target Debugging	
Support for Debugging Multiple Processors from Single Terminal	
Customization of Debugger for New Target Environment	
Symbolic Debugging	
Tracing	
Breakpoint Control	
Data Flow Tracing	
Path Flow Tracing	
Selectable Level Of Granularity	
Display	
Program Source	
History	
Stack (Calling Hierarchy)	
Tasks	
Rendezvous Status	
Breakpoints	
Tracepoints	
Memory	
Collections And Global Heaps	
Name Of Current Exception	
Evaluate Objects	
Step	
Single	
By Discrete Amounts	
Into Subprograms	
Over Subprograms	
To Next Scheduling Event	
To Next Exception	
To End of Program Unit	
Miscellaneous	
Symbol Abbreviation	
Set Context For Control	
Input Debugger Command Files	
Modify Variable Values	
Modify Object Code	
Modify Control Flow	
Console Interrupt	
Full Screen Mode	
Keypad For Entering Commands	
Virtual Clock	
Special Compilation Mode	
Multi-Language Support	
Complete Ada Language Support with Deep Nesting	
Dynamic Interrupt	
Optimization Support	
Units Comprising Executable	
Locate Objects with Overloaded Names	
No Overhead to Explicitly Create a Debug File	

6.6 TIMING ANALYSIS CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of timing analysis by developing a list of functional capabilities.
[@RM: Timing Analysis 7.3.2.14, @RM: Completeness 6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 6.6-1) and timing analysis system documentation.

Process: Check off capabilities demonstrated by the timing analysis system or discussed in the documentation.

Outputs: A list of capabilities provided by the timing analysis system.

Table 6.6-1 Timing Analysis Capabilities Checklist

FEATURE	FOUND
Timing Instrumentation	
Intrusive	
Non-Intrusive	
User Specified Error Tolerances	
Use of Timing Loop	
Repetitive Execution Until Stable Convergence	
Measurement of Overhead Execution	
Test for Clock Jitter	
System Clock Accuracy Consideration	
Hardware Organization (Cache, Pipeline ...) Considerations	
Operating System (Virtual, Multiprocessing ...) Considerations	
Size of Test Problem Considerations	
Fraction By Section Of Code	
Tasking Monitor	
Fraction Executing	
Fraction Runnable	
Fraction Runnable and not Executing	
Time Between Runnable and Executing	
Time Between Events	
System Idle Time	
Miscellaneous	
Timing Loop Code is System Independent	
Special Hardware is not Needed	
Code to be Measured is Easily Installed	
Output Shows Variations in Measurements	
Statistical Measurements are Available	
Use of Computer Resources is Minimized	
Measures Either Wall or Clock Time	

6.7 REAL-TIME ANALYSIS CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of real-time analysis by developing a list of functional capabilities.
[@RM: Real-Time Analysis 7.3.2.17, @RM: Completeness 6.4.9]

Primary References:
[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.
[Classification Schema/E&V Taxonomy Checklists: 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 6.7-1) and real-time analysis system documentation.

Process: Check off capabilities demonstrated by the real-time analysis system or discussed in the documentation.

Outputs: A list of capabilities provided by the real-time analysis system.

Table 6.7-1 Real-time Analysis Capabilities Checklist

FEATURE	FOUND
Hardware-In-The-Loop Non-Intrusive Instrumentation Performance Analysis Symbolic Trace	

6.8 INSTRUCTION-LEVEL SIMULATION CHECKLIST

Purpose: Evaluation of the completeness of instruction-level simulation by developing a list of functional capabilities
[@RM: Simulation and Modeling 7.3.2.3, @RM: Completeness 6.4.9]

Primary References:
[Weiderman 1987b] N.H. Weiderman, et al., "Ada for Embedded Systems: Issues and Questions," Software Engineering Institute, Technical Report CMU/SEI-87-TR-26, December 1987, DTIC Number AD A191 096.

Vendors/Agents: [SEI]

Method: Checklist.

Inputs: Capabilities checklist (see Table 6.8-1) and instruction-level simulation system documentation.

Process: Check off capabilities demonstrated by the instruction-level simulation system or discussed in the documentation.

Outputs: A list of capabilities provided by the instruction-level simulation system.

Table 6.8-1 Instruction-level Simulation Checklist

FEATURE	FOUND
Accurately simulates both the functional and temporal behavior of the target's instruction set architecture	
Provides access to all memory locations and registers	
Supports typical features found in a symbolic debugger	
Single-step instruction execution	
Examines variable values	
Start/stop program execution	
Performs timing analysis	
Provides assembler instruction execution times	
Provides Ada instruction execution times	
Provides Ada subprogram execution times	
Supports simulated input/output interaction	
Provides access to I/O ports	
Provides access to device control and data registers	
Emulates the architecture's interrupt mechanism	
Facilitates the set-up and reuse of test sessions	
Freezes the current session's context	
Executes debugger commands from script files	
Supplies I/O data from existing data files	

7. TEST SYSTEMS ASSESSORS

These assessors examine the ability of the APSE or APSE component to support and facilitate the planning, development, execution, evaluation, and documentation of tests of software.

7.1 TESTING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness and power of testing by developing a list of functional capabilities.
[@RM: Analysis 7.3, @RM: Completeness 6.4.9; @RM: Power 6.4.21]

Primary References:
[DeMillo 1986] R.A. DeMillo, "Functional Capabilities of a Test and Evaluation Subenvironment in an Advanced Software Engineering Environment," Georgia Institute of Technology GIT-SERC-86/07, 20 October 1986.

Vendors/Agents: [GIT]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 7.1-1) and testing system documentation.

Process: Check off capabilities demonstrated by the testing system or discussed in the documentation.

Outputs: A list of capabilities provided by the testing system.

Table 7.1-1 Testing Capabilities Checklist

FEATURE	FOUND
Static Analyzers Code Auditors Consistency Checkers Interface Analyzers Completeness Checkers Tool Building Services Common "Front-End" Facilities for Languages of Interest (Parsing, Source & Internal Form Manipulation, Execution Facilities) Tool Composition Aids Test Building Services (including Test Data Generators) Symbolic Evaluators Component Coverage Analyzers Data Flow Analyzers Assertion Processors Mutation Analyzers Path and Domain Selection Aids Random Test Generators Test Description and Preparation Services Data Editors Data Auditors Body/Stub Generators File Comparators Data/File Services Software and System Test Communications Facilities Test Execution Services Test Harness Generator Data and Error Logging Services Quality Measurement Tools Test Analysis Services Correctness Analyzers (Oracles) Instrumentation Aids Status Display Tools Data Reduction and Analysis Tools Cross Reference (Traceability) Management and Analysis Tools Decision Support Services Documentation Services Information Repositories Problem Report Processing and Analysis Tools Change Request Processing and Analysis Tools	

7.2 SEI UNIT TESTING AND DEBUGGING EXPERIMENT

Purpose: Evaluation of an environment's capabilities, from the point of view of the unit tester. An experiment was designed to simulate the activities normally associated with small projects, namely the design, creation, modification, and testing of a single unit or module. See also the SEI Design Support Experiment [9.1].

[@RM:	Debugging	7.3.2.5,	@RM:	Power	6.4.21;
@RM:	Debugging	7.3.2.5,	@RM:	Processing Effectiveness	6.4.22;
@RM:	Dynamic Analysis	7.3.2,	@RM:	Power	6.4.21;
@RM:	Dynamic Analysis	7.3.2,	@RM:	Processing Effectiveness	6.4.22]

Primary References:

[Weiderman 1987] N. Weiderman and N. Haberman, "Evaluation of Ada Environments," *Software Engineering Institute*, Technical Report CMU/SEI-87-TR-1, March 1987, Chapter 6, DTIC Number AD A180 905.

[Weiderman: Evaluation of Ada Environments, 4.13]

Host/OS: VAX/VMS and VAX/UNIX

Vendors/Agents: [SEI]

Method: Structured experiment

Inputs: The "generic" experiment description, an APSE, and host (and target) computer.

Process: "Instantiate" the experiment for a specific Host/OS/APSE combination and carry it out.

Outputs: A filled-in table of functional elements present and missing, elapsed time and cpu time values, and subjective judgments based on the experience.

8. TOOL SUPPORT COMPONENT ASSESSORS

These assessors evaluate or validate implementations of specifications for tool support components. Components that may be assessed could include a CAIS or a CAIS-A implementation, Portable Common Tool Environment (PCTE) implementations, and Ada language interfaces to the UNIX operating system and its variants (e.g., Berkeley UNIX, System V, A/UX, POSIX). Also included here are window managers (such as X-windows), language bindings to standard interface specification implementations (such as Ada bindings to GKS or SQL), I/O pipes, and RAM cache.

8.1 CAIS IMPLEMENTATION VALIDATION CAPABILITY (CIVC)

Purpose: Assess the conformance of CAIS implementations to the DoD-STD-1838 standard. The CIVC consists of a test suite, analysis tools, and associated documentation which enable validators and CAIS implementors to determine the completeness of CAIS implementations with respect to conformance to the standards. The test suite consists of tests to be compiled and executed with interfaces provided for a CAIS implementation. Analysis tools are utilized for aiding the users in selecting tests and obtaining results.
[@RM: Kernel 7.2.3.3, @RM: Completeness 6.4.9]

Primary References: [CIVC 1989] "CIVC1 Implementor's Guide," Air Force Wright Aeronautical Laboratory, CIVC-FINL-019, October 1989, in progress.

Vendors/Agents: TBA

Method: Automated test suite

Inputs: The CIVC test suite, CAIS implementation, Ada compiler and runtime system, and host computer.

Process:

1. Obtain the CIVC test suite
2. Compile and run the tests
3. Collect and analyze the results.

Outputs: Report describing the conformance of various aspects of the CAIS implementation to DoD-STD-1838.

8.2 TOOL SUPPORT INTERFACE EVALUATION

Purpose: Evaluation of tool support interfaces in terms of four criteria: level, appropriateness, implementability, and performance. Five "scenarios" were designed and used to exercise a prototype CAIS implementation and a prototype PCTE implementation. The scenarios involved a configuration management system, an edit-compile-link-test cycle, a conference management system, a window manager, and a design editor.
[@RM: Kernel 7.2.3.3]

Primary References:
[Long 1988] F.W. Long, and M.D. Tedd, "Evaluating Tool Support Interfaces," Ada in Industry, Proceedings of the Ada-Europe Conference, Munich, 7-9 June 1988, Cambridge University Press, 1988.

Host/OS: Sun

Vendors/Agents: [College of Wales, UK]

Method: Structured experiment

Inputs: The source code for the scenarios, the tool support interface(s) (CAIS, PCTE, other), Ada compiler and runtime system, and host computer.

Process:

1. Obtain the source code for the scenarios
2. Compile and run the scenario(s)
3. Collect the results.

Outputs: Objective results and subjective conclusions concerning the impact on tool writers and the cost and behavior of the interface implementation.

9. REQUIREMENTS/DESIGN SUPPORT ASSESSORS

These assessors measure the suitability and effectiveness of various software definitic.. specification, and design tools. This specifically includes assessors of Ada Program Design Language (PDL) implementations and/or guidelines in the use of Ada as a PDL.

9.1 SEI DESIGN SUPPORT EXPERIMENT

Purpose: Evaluation of the design and code development capabilities of an environment, as represented in a small project. An experiment was designed to simulate the activities normally associated with small projects, namely the design, creation, modification, and testing of a single unit or module. See also the SEI Unit Testing and Debugging Experiment [7.2].

[@RM: Preliminary Design 7.1.6.4, @RM: Power 6.4.21;
@RM: Preliminary Design 7.1.6.4, @RM: Processing Effectiveness 6.4.22;
@RM: Preliminary Design 7.1.6.4, @RM: Storage Effectiveness 6.4.31;
@RM: Detailed Design 7.1.6.5, @RM: Power 6.4.21;
@RM: Detailed Design 7.1.6.5, @RM: Processing Effectiveness 6.4.22;
@RM: Detailed Design 7.1.6.5, @RM: Storage Effectiveness 6.4.31]

Primary References:

[Weiderman 1987] N. Weiderman and N. Haberman, "Evaluation of Ada Environments," Software Engineering Institute, Technical Report CMU/SEI-87-TR-1, March 1987, Chapter 5, DTIC Number AD A180 905.

[Weiderman: Evaluation of Ada Environments, 4.13]

Host/OS: VAX/VMS and VAX/UNIX

Vendors/Agents: [SEI]

Method: Structured experiment.

Inputs: The "generic" experiment description, an APSE, and host computer.

Process: "Instantiate" the experiment for a specific Host/OS/APSE combination and carry it out.

Outputs: A filled-in checklist of functional elements present and missing, tables of time and space data, and subjective judgments based on the experience.

9.2 REQUIREMENTS PROTOTYPING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of requirements prototyping by developing a list of functional capabilities.
[@RM: Requirements Prototyping 7.3.2.2, @RM: Completeness 6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 9.2-1) and requirements prototyping documentation.

Process: Check off capabilities demonstrated by the requirements prototyping system or discussed in the documentation.

Outputs: A list of capabilities provided by the requirements prototyping system.

Table 9.2-1 Requirements Prototyping Capabilities Checklist

FEATURE	FOUND
Standards Requirements Libraries Executable Specifications Fourth Generation Languages or Very High Level Languages Reusable Building Blocks and Associated Tools Man-Machine Interface Prototyping Capabilities Applications Generators Previous Software Version Import Capabilities	

9.3 SIMULATION AND MODELING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of simulation and modeling by developing a list of functional capabilities.
[@RM: Simulation and Modeling 7.3.2.3, @RM: Completeness 6.4.9]

Primary References:
[ISTAR 1987] Workshop on Future Development Environments, Information Science and Technology Assessment for Research, Conference on Information Mission Area (IMA) Productivity, Department of Army Director of Information Systems for Command, Control, Communications and Computers, 13-15 April 1987, pp 28.

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 9.3-1) and simulation and modeling documentation.

Process: Check off capabilities demonstrated by the simulation and modeling system or discussed in the documentation.

Outputs: A list of capabilities provided by the simulation and modeling system.

Table 9.3-1 Simulation and Modeling Capabilities Checklist

FEATURE	FOUND
Conceptual Modeling Support	
Domain-Specific Knowledge Base	
Inferencing Systems	
Operational Environment Modeling Support	
User Modeling Support	
Model Browsers	
Game and Risk Models Database	
Functional Allocation Methodologies Database	
Scaling Rules Database	
Constraint Evaluation Tools	
Precision Estimators	
Computer System Modeling	
Interface/Input Support	
Graphical	
Menus	
Tabular	
Command	
Model Subject	
Control Flow	
Information Flow	
Human/Machine Procedures	
Outputs	
Response Time Estimates	
Throughput Estimates	
Resource Utilization Estimates	
System Types Supported	
Real Time	
Distributed	
Multiprocessor	
Standard Computer System Models Database	
Preprogrammed Models of Common Distributed System Functions	
Underlying Mathematical Theory (e.g., Queueing Network Theory)	

9.4 NADC/SPS CASE TOOLS EVALUATION

Purpose: Development of evaluation criteria and evaluation of selected candidate methods and tools. The focus of this investigation was Computer Aided Software Engineering (CASE) tools and methods applied during the early life cycle phases/activities (system and software requirements and top-level design) and applied to large, time-critical systems. Three commercially available CASE tools were selected for evaluation, following an initial survey of more than 100 possibilities. An experiment based on a sample problem (submarine detection with a sonobouy) was created and carried out using three systems. Evaluation criteria, detailed questions, and a scoring system were developed and applied in three areas: method, automation, and vendor support.

[@RM: System Requirements 7.1.6.1, @RM: Augmentability 6.4.4;
@RM: Software Requirements 7.1.6.2, @RM: Augmentability 6.4.4;
@RM: Preliminary Design 7.1.6.4, @RM: Augmentability 6.4.4;
@RM: System Requirements 7.1.6.1, @RM: Capacity 6.4.6;
@RM: Software Requirements 7.1.6.2, @RM: Capacity 6.4.6;
@RM: Preliminary Design 7.1.6.4, @RM: Capacity 6.4.6;
@RM: System Requirements 7.1.6.1, @RM: Completeness 6.4.9;
@RM: Software Requirements 7.1.6.2, @RM: Completeness 6.4.9;
@RM: Preliminary Design 7.1.6.4, @RM: Completeness 6.4.9;
@RM: System Requirements 7.1.6.1, @RM: Processing Effectiveness 6.4.22;
@RM: Software Requirements 7.1.6.2, @RM: Processing Effectiveness 6.4.22;
@RM: Preliminary Design 7.1.6.4, @RM: Processing Effectiveness 6.4.22;
@RM: System Requirements 7.1.6.1, @RM: Self Descriptiveness 6.4.28;
@RM: Software Requirements 7.1.6.2, @RM: Self Descriptiveness 6.4.28;
@RM: Preliminary Design 7.1.6.4, @RM: Self Descriptiveness 6.4.28;
@RM: System Requirements 7.1.6.1, @RM: System Clarity 6.4.33;
@RM: Software Requirements 7.1.6.2, @RM: System Clarity 6.4.33;
@RM: Preliminary Design 7.1.6.4, @RM: System Clarity 6.4.33;
@RM: System Requirements 7.1.6.1, @RM: Training 6.4.36;
@RM: Software Requirements 7.1.6.2, @RM: Training 6.4.36;
@RM: Preliminary Design 7.1.6.4, @RM: Training 6.4.36]

Primary References:

[Donaldson 1988] C. Donaldson and P.B. Dyson, "Computer-Aided Systems and Software Engineering Products for Time-Critical Applications Development," Software Productivity Solutions (SPS), Inc., April 1988.

[Stuebing 1988] H.G. Stuebing, "Evaluation of Computer Aided Systems/Software Engineering Products for Time-Critical Naval Systems," Proceedings of the Conference *Methodologies and Tools for Real Time Systems*, November 14-15, 1988.

Host/OS: Various Workstations: PC/AT, Apollo, Sun, VAXStation

Vendors/Agents: [SPS]

Method: Structured Experiment

Inputs: Evaluation criteria and questions, sample problem definition, and candidate methods and tools

Process: For each candidate method/tool, carry out development of software requirement specification and top-level design documents. Answer evaluation questions and fill out scoring sheets.

Outputs: Evaluation reports containing two levels of detail: executive summaries with top-level scoring and detailed descriptions and analyses with questions, answers, and individual scores.

9.5 TIME-CRITICAL APPLICATIONS SUPPORT CHECKLIST

Purpose: Evaluation of the completeness of time-critical applications support by developing a list of functional capabilities.
[@RM: Systems Requirements 7.1.6.1, @RM: Completeness 6.4.9;
@RM: Software Requirements 7.1.6.2, @RM: Completeness 6.4.9;
@RM: Preliminary Design 7.1.6.4, @RM: Completeness 6.4.9;
@RM: Detailed Design 7.1.6.5, @RM: Completeness 6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Checklist.

Inputs: Capabilities checklist (see Table 9.5-1) and time-critical applications support documentation.

Process: Check off capabilities demonstrated by the time-critical applications support or discussed in the documentation.

Outputs: A list of capabilities provided by the time-critical applications support.

Table 9.5-1 Time-critical Applications Support Checklist

FEATURE	FOUND
Periodic and aperiodic events	
Synchronization of sequential and concurrent processes	
Execution sequence of components	
Timing constraints for events, sequences of events, processes, and sequences of processes	
Precision of a system's response to internal and external events	
Interrupts and the extent of process context switching	
Processing of discrete and time continuous data	
Allocation of critical timelines and resource utilizations	
Data throughput	
Task priority changes due to system mode changes or failures	
Task management (time-slicing, run-to-completion)	
Graphical time-line depiction for tasks, showing dependencies and concurrencies	
Simulations for the host	
Dynamic analysis for the target	

10. CONFIGURATION MANAGEMENT SUPPORT ASSESSORS

These assessors examine the performance, usability, and completeness of the APSE or APSE component functionality related to controlling the contents of software systems. This includes monitoring the status, preserving the integrity of released and developing versions, and controlling the effects of changes throughout the lifetime of the software system.

10.1 CONFIGURATION MANAGEMENT CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness of configuration management by developing a list of functional capabilities.
[@RM: Configuration Management 7.2.2.7, @RM: Completeness 6.4.9]

Primary References:
[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.
[Classification Schema/E&V Taxonomy Checklists: 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 10.1-1) and configuration management documentation.

Process: Check off capabilities demonstrated by the configuration manager or discussed in the documentation.

Outputs: A list of capabilities provided by the configuration manager.

Table 10.1-1 Configuration Management Capabilities Checklist

FEATURE	FOUND
Version Management	
Archive	
Protect	
Revision Management	
Support for Multiple Development Paths	
Audit Support	
Configuration Library	
Create	
Delete	
Verify	
Library Elements	
Create	
Delete	
Fetch	
Reserve	
Unreserve	
Replace	
Differences	
Element Classes	
Create	
Delete	
Insert Element	
Remove Element	
Listings	
Elements	
Reservation	
History	
Annotation	
Completeness	
Level Control	
Usage Administration	
Test Control	
Procedures	
Data	
Results	
Failure Reporting	
Activity Tracking	
Integration with Development Environment	

10.2 SEI CONFIGURATION MANAGEMENT EXPERIMENT

Purpose: Evaluation of the configuration management and version control capabilities of an environment. An experiment was designed to simulate the system integration and testing phase of the life cycle by having three separate development lines of descent from a single baseline.
[@RM: Configuration Management 7.2.2.7, @RM: Power 6.4.21;
@RM: Processing Effectiveness 6.4.22]

Primary References:
[Weiderman 1987] N. Weiderman and N. Haberman, "Evaluation of Ada Environments," Software Engineering Institute, Technical Report CMU/SEI-87-TR-1, March 1987, Chapter 3, DTIC Number AD A180 905.
[Weiderman: Evaluation of Ada Environments, 4.13]

Host/OS: VAX/VMS and VAX/UNIX

Vendors/Agents: [SEI]

Method: Structured experiment.

Inputs: The "generic" experiment description, an APSE, and host computer.

Process: "Instantiate" the experiment for a specific Host/OS/APSE combination and carry it out.

Outputs: A filled-in checklist showing functional elements present and missing, a table of elapsed-time values for certain specific operations, and subjective judgments based on the experience.

10.3 CONFIGURATION MANAGEMENT ASSESSMENT QUESTIONNAIRE

Purpose: The document presents a hierarchical breakdown of configuration management shown in Fig. 10.3-1. Requirements for each element in the hierarchy are listed for certain attributes. Each requirement is augmented by one or more questions which address the requirement.
[@RM: Configuration Management 7.2.2.7, @RM: Attributes 6.]

Primary References:
[E&V Report 1984] "Requirements for Evaluation and Validation of Ada Programming Support Environments, Version 1.0," 17 October 1984, Appendix B of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, November 1984, pp. B-86 - B-91, DTIC Number AD A153 609.
[Requirements for E&V: 4.5]

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire and configuration management documentation.

Process: Answer questions based on documentation, using the configuration manager, or asking the vendor.

Outputs: Completed questionnaire.

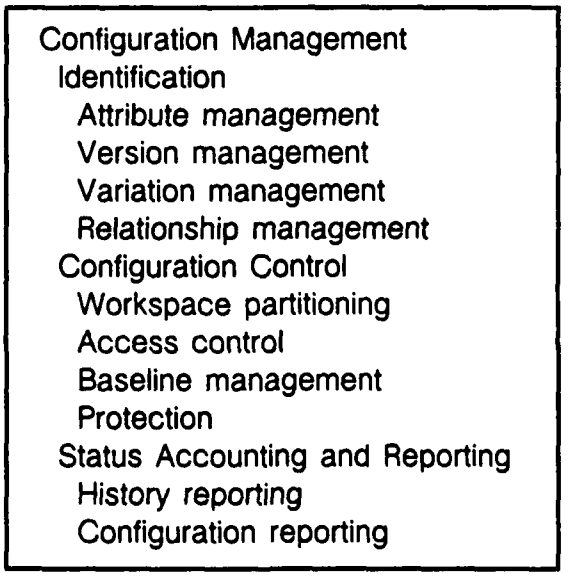

Figure 10.3-1 Configuration Management Hierarchy

11. DISTRIBUTED SYSTEMS DEVELOPMENT AND RUNTIME SUPPORT ASSESSORS

These assessors examine the ability of the APSE or APSE components to support software development for distributed processing systems, and to provide runtime support for distributed processing systems.

12. DISTRIBUTED APSE ASSESSORS

These assessors examine the ability of two or more distributed APSEs to communicate in cooperative ways in supporting the development of mission critical software at diverse geographical locations.

12.1 DISTRIBUTED APSE QUESTIONNAIRE

Purpose: Evaluation of the APSE's ability to be used in a distributed environment.
[@RM: Whole APSE Assessment Issues 3.; @RM: Commonality 6.4.7;
@RM: Consistency 6.4.10; Functional Overlap 6.4.14;
@RM: Operability 6.4.20; @RM: Required Configuration 6.4.26;
@RM: System Compatibility 6.4.34]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire (see Fig. 12.1-1) and APSE documentation.

Process: Answer questions based on documentation, using the APSE, or asking the vendor.

Outputs: Completed questionnaire.

Host Transparency
Is the functionality of the APSE the same across all nodes?

Heterogenous/Homogenous
Does the APSE support a heterogenous hardware configuration or is it restricted to implementation on a homogenous hardware configuration?
- Is there special hardware required for its implementation on a heterogenous configuration?
- Are there special software communication protocols that are required for implementation on a heterogenous configuration?

Data Availability
Are all APSE data available to all APSE nodes or tools?

Common User Interface
Can the user access and use the APSE from any node without retraining or using a different set of commands?

Figure 12.1-1 Distributed APSE Questionnaire

13. "WHOLE APSE" ASSESSORS

These assessors examine or measure the overall quality or performance of an APSE considered as a whole rather than as a collection of individual parts individually assessed. A specific whole-APSE assessor may be designed to achieve a limited objective. An example of a limited objective is: evaluate the quality of an APSE in supporting a team of software developers performing a specific life cycle phase or activity such as preliminary design or integration testing. The results of such an evaluation could then become one ingredient of an integrated whole-APSE assessment (as described in Section 3.3), which has a broad objective.

13.1 APSE CHARACTERIZATION

Purpose: The purpose of this form is to provide an overview or summary of the capabilities and features of an APSE. This form can be used as an initial information gathering device to begin the process of whole-APSE assessment. This information would then be supplemented by results of detailed evaluations or examinations of attributes that are of specific interest to the potential buyer or user of an APSE.
[@RM: Whole APSE Assessment Issues 3., @RM: Capacity 6.4.6; @RM: Completeness 6.4.9; @RM: Cost 6.4.11; @RM: Maturity 6.4.18; @RM: Operability 6.4.20; @RM: Power 6.4.21; @RM: Required Configuration 6.4.26]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Blank APSE characterization form (see Fig. 13.1-1) and APSE documentation.

Process:

1. Complete the APSE characterization form
2. Select APSEs for further investigation based on information gathered from step 1.

Outputs: Completed APSE characterization form.

Name/Acronym:

Vendor:

Address:

Phone Number:

Cost ($, no charge, not available/applicable):

Purchase	_____	Seminars	_____
Maintenance	_____	In House Classes	_____
Documentation	_____	Educational Videos	_____
On-Line Help	_____	On-Line Tutorials	_____
Hot-Line Support	_____		

Problem Reporting/Resolution Procedures:

Frequency of Updates:

Usage Limitations (License Restrictions):

Host/Target(s) — Required Configurations:

Peripherals Supported:

Languages Supported & Interoperability Features:

Summary of Features:

Life Cycle Support — Capabilities/Major Activity:

Methodology Support:

Management Support:

Application-Specific Capabilities:

Documentation Support (editors, word processors, document generators, desktop publishing):

Figure 13.1-1 APSE Characterization Form

File/Database/Program Library Management (hierarchical, relational):

Access Control — Level of Granularity:

Integration Mechanism (standard file structures, database, standard intertool interfaces):

User Interface (command language, menus, icons) — Flexibility vs. Consistency:

Extensibility:

Support for Distributed Development:

Capacity (number of users, size of project):

Typical Usage Scenarios (expertise of users, roles):

Developer:

Production Process/Vehicles:

Date First Released:

Previous Use:

References (documentation, evaluation results, case histories):

Figure 13.1-1 APSE Characterization Form (Continued)

13.2 Ada-EUROPE Ada ENVIRONMENT QUESTIONNAIRES

Purpose: The Ada-Europe Environment Working Group, under the chairmanship of John Nissen, produced a guide which adopts the "point of view of a potential user wishing to select an environment, and provides lists of questions to be asked about the environment under consideration." It generally follows the structure proposed in Stoneman [@DoD 1980]; it "starts from the inside of the onion structure and works outwards." Each of its 19 chapters follows a standard format. Topics are introduced and discussed, typically using one or two pages of text, and then a list of appropriate questions is provided.

[@RM: Whole APSE Assessment Issues 3., @RM: Augmentability 6.4.4; @RM: Capacity 6.4.6; @RM: Commonality 6.4.7; @RM: Completeness 6.4.9; @RM: Operability 6.4.20; @RM: Power 6.4.21; @RM: Processing Effectiveness 6.4.22; @RM: Proprietary Rights 6.4.23; @RM: Required Configuration 6.4.26; @RM: System Availability 6.4.32; @RM: Training 6.4.36]

Primary References:

[Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986.

[Ada-Europe: Selecting an Ada Environment: 4.9]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaires

Inputs: Questionnaires (see Table 13.2-1), APSE, and APSE documentation.

Process: Answer the questions by using the APSE, reading the documentation, or asking the vendor of the APSE.

Outputs: Completed questionnaires.

Table 13.2-1 Ada-Europe Environment Questionnaires

ATTRIBUTE	RELEVANT SECTION(S) FROM LYONS BOOK
Augmentability	7.2, 7.4, 9.
Capacity	18.1
Commonality	8.
Completeness	4., 5., 7.1, 7.3, 9., 13.-17.
Operability	10.-12., 18.10-18.12
Power	10.3-10.6
Processing Effectiveness	18.2-18.19
Proprietary Rights	19.
Required Configuration	2., 3., 18.2, 18.3
System Accessibility	6.
Training	10.2

13.3 CROSS DEVELOPMENT SYSTEM SUPPORT QUESTIONNAIRE

Purpose: Evaluation of the APSE's ability to support the development of an application on a host computer for implementation on a *different* target computer, where the target computer is usually incapable of compiling, linking, and debugging software.

[@RM: Whole APSE Assessment Issues 3.,

@RM: Assembling	7.1.6.6,	@RM: Completeness 6.4.9;
@RM: Compilation	7.1.6.7,	@RM: Completeness 6.4.9;
@RM: Linking/Loading	7.1.6.13,	@RM: Completeness 6.4.9;
@RM: Simulation and Modeling	7.3.2.3,	@RM: Completeness 6.4.9;
@RM: Debugging	7.3.2.5,	@RM: Completeness 6.4.9;
@RM: Emulation	7.3.2.13,	@RM: Completeness 6.4.9;
@RM: Timing Analysis	7.3.2.14,	@RM: Completeness 6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire (see Fig. 13.3-1) and APSE documentation.

Process: Answer questions based on documentation, using the tools, or asking the vendor.

Outputs: Completed questionnaire.

Transformation

Are there target-optimizing cross-assemblers?
Does the front end support multiple code generators?
What language features are supported by the code generator?
Where are the pragmas defined?
- Are they all defined and understood by the front end?
- Are they all defined in the front end, but some are understood in the front end and some are understood in the back end?

Does it provide the same pragma support across all code generators?
Are there conditional compilation capabilities?
What is the extent of the target features which are supported:
- 1750A (timer a, timer b, extended memory, etc.)
- 68020 (whatever particular features are identified by this chip)
- 80960 (whatever particular features are identified by this chip)

Is there an intelligent, modifiable linker?

Analysis

Is there a host-based target emulator, simulator, and symbolic debugger?
Is there a facility for supporting interoperability (communcations paths) between simulated target processors for multi-target debugging?
Does the host development system have visibility into actual target processor hardware during execution?
- If so, is such visibility in terms of original source code names?
- If so, is such visibility extendable into multiple targets?

Is there a host-based static target timing analysis capability?

Figure 13.3-1 Cross Development System Support Questionnaire

13.4 APSE CUSTOMIZATION QUESTIONNAIRE

Purpose: Evaluation of the APSE's ability to be customized for a particular host and target environment, methodology, or application domain.
[@RM: Whole APSE Assessment Issues 3.,
@RM: Predefined and User-Defined Forms 7.1.2.3;
@RM: Database Management 7.2.1.1; @RM: Documentation Management 7.2.1.2;
@RM: Configuration Management 7.2.2.7; @RM: Runtime Environment 7.2.3.5;
@RM: Application Independence 6.4.3; @RM: Commonality 6.4.7;
@RM: Distributeness 6.4.12; @RM: Generality 6.4.16;
@RM: Modularity 6.4.19; @RM: Operability 6.4.20;
@RM: Rehostability 6.4.25; @RM: Required Configuration 6.4.26;
@RM: Retargetability 6.4.27; @RM: System Compatibility 6.4.34]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire (see Fig. 13.4-1) and APSE documentation.

Process: Answer questions based on documentation, using the tools, or asking the vendor.

Outputs: Completed questionnaire.

Methodology
- Can the user tailor the methodology supported to fit his own needs, such as for rapid prototyping or partial life cycle completion?
- Can the user define his own methodology?

Automation
- Is there development automation present in the APSE?
- — If so, can it be modified to reflect:
  - — different project management organization?
  - — different life cycle definition (standard or user-defined)?
  - — different document standards generation?
  - — different project-specific configuration management?

Documentation
- Is there documentation support for:
- — user-defined document formats?
- — customer-defined document formats?
- Can the configuration management for documentation be altered?
- Does the APSE support the planning, design, generation, baseline, and maintenance of documents?
- Is the information created by the APSE directly importable into the documents?
- Can information from one document be transferred to another?
- Are changes to the system automatically reflected in the system documentation?
- Does the APSE support merged text and graphics documentation?
- Is the documentation resident in one database or is it derived from multiple databases?
- Is the documentation exportable to another APSE or another database?

User Role Change
- Can the APSE be modified to support a user in:
- — a different skill level?
- — a different job assignment?

Communication
- What communication protocols are supported by the APSE?
- — Can they be modified?
- — Can they be user-defined?
- — Is special hardware required to support this communication?

Distribution
- Can the APSE go from a single host to support multiple homogenous hosts? Heterogenous hosts?
- Can the APSE go from an homogenous to an heterogenous environment and vice versa?

Host Dependencies
- Can the APSE be modified for use on another host?
- Is the APSE built on top of a portability interface implementation such as the CAIS?

Target Dependencies
- Does the APSE support one target?
- — Can it be modified?
- — Can other targets be supported?
- Can multiple targets be supported at one time or only one target?
- Does the APSE support real-time embedded or non real-time, embedded targets only?

Runtime Support System
- Can the RTS be modified?
- Are there standard modifications (versions) provided?
- Does it have a modular construction
- Is the design documentation for it provided?
- — Is it easily understood?
- Is the associated toolset (linker, loader, compiler) modifiable to support the modifications of the RTS?
- Is the RTS source code provided?
- Does the RTS support multiple targets?

Figure 13.4-1 APSE Customization Questionnaire

14. ADAPTATION ASSESSORS

These assessors examine the ease with which an APSE or APSE component can be used beyond its original requirements, such as extending or expanding capabilities and adapting for use in another application or environment. This is measured as the degree to which this adaptation can be accomplished without reprogramming.

14.1 HOST AND TARGET QUESTIONNAIRE

Purpose: Evaluation of tools relative to host and target configurations.
[@RM: Rehostability 6.4.25;
@RM: Retargetability 6.4.27]

Primary References:
[Nissen 1984] J.C.D. Nissen, B.A. Wichman, et al., "Guidelines for Ada Compiler Specification and Selection," in Ada: *Language, Compilers and Bibliography*, ed. M.W. Rogers, Cambridge University Press, 1984.
[Nissen, et al.: Guidelines For Ada Compiler Specification And Selection: 4.19]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaire

Inputs: Questionnaire and tool documentation.

Process: Fill in the appropriate answers in the following questionnaire.

a) Host configuration(s) required
b) Host operating system(s) required
c) Target configuration(s) supported
d) Target operating system(s) supported
e) APSE(s) supported, if applicable
f) Host-target communication supported
    i) program loading
    ii) program execution and debugging.

Outputs: A completed list which characterizes the tool relative to host-target issues.

14.2 MACHINE-SPECIFIC CHARACTERISTICS QUESTIONNAIRE

Purpose: Evaluation of tools relative to machine-specific characteristics.
[@RM: Rehostability 6.4.25;
@RM: Retargetability 6.4.27]

Primary References:
[Nissen 1984] J.C.D. Nissen, B.A. Wichman, et al., "Guidelines for Ada Compiler Specification and Selection," in Ada: *Language, Compilers and Bibliography*, ed. M.W. Rogers, Cambridge University Press, 1984.
[Nissen, et al.: Guidelines For Ada Compiler Specification And Selection: 4.19]

Vendors/Agents: [Cambridge University Press]

Method: Questionnaire

Inputs: Questionnaire and tool documentation.

Process: Fill in the appropriate answers in the following questionnaire.

[@DoD 1983: Lexical Elements 2.]

- [@DoD 1983: 2.1] Character set of the host and target
- [@DoD 1983: 2.2] Maximum number of characters on a line of the host and target
- [@DoD 1983: 2.3, 2.4] Is the maximum character length of an identifier or numerical literal restricted other than by line length
- [@DoD 1983: 2.8, F.] The form, allowed place, and effect of every implementation-defined pragma

[@DoD 1983: Declarations and Types 3.]

- [@DoD 1983: 3.2.1] The effect of using uninitialized variables — does the compiler flag or reject program that depends upon such variables
- [@DoD 1983: 3.5.1] The maximum number of elements in an enumeration type
- [@DoD 1983: 3.5.4] The values of:
  - -- INTEGER'FIRST
  - -- SHORT_INTEGER'FIRST
  - -- LONG_INTEGER'FIRST
  - -- INTEGER'LAST

-- SHORT_INTEGER'LAST
-- LONG_INTEGER'LAST

- [@DoD 1983: 3.5.8] The values of:
  -- FLOAT'DIGITS
  -- SHORT_FLOAT'DIGITS
  -- LONG_FLOAT'DIGITS

[@DoD 1983: Names and Expressions 4.]

- [@DoD 1983: 4.10] Is there a limit on the range of universal values which exceeds the capacity of the compiler
- [@DoD 1983: 4.10] Is there a limit on the accuracy real universal expressions

[@DoD 1983: Tasks 9.]

- [@DoD 1983: 9.6] The values of:
  -- DURATION'DELTA
  -- DURATION'SMALL
  -- DURATION'FIRST
  -- DURATION'LAST
- [@DoD 1983: 9.8] The values of:
  -- PRIORITY'FIRST
  -- PRIORITY'LAST
- [@DoD 1983: 9.11] The restrictions on shared variables

[@DoD 1983: Program Structure and Compilation Issues 10.]

- [@DoD 1983: 10.1] Initiation, communication with, and restrictions on the main program
- [@DoD 1983: 10.5] When tasks initiated in imported library units will terminate

[@DoD 1983: Exceptions 11.]

- [@DoD 1983: 11.1] Conditions under which these exceptions are raised:
  -- NUMERIC_ERROR
  -- PROGRAM_ERROR
  -- STORAGE_ERROR

[@DoD 1983: Representation Clauses and Implementation-Dependent Features 13.]

- [@DoD 1983: 13.4, F.] The list of all restrictions on representation clauses
- [@DoD 1983: 13.1, F.] The conventions used for any system generated name denoting system dependent components
- [@DoD 1983: 13.5, F.] The interpretation of expressions that appear in address clauses, including those for interrupts
- [@DoD 1983: 13.7] The specification of package SYSTEM; which includes the values of:
  - -- MIN_INT
  - -- MAX_INT
  - -- MAX_DIGITS
  - -- MAX_MANTISSA
  - -- FINE_DELTA
  - -- TICK

[@DoD 1983: 13.7.3] For a pre-defined floating point type F, the value of:

  - -- F'MACHINE_ROUNDS
  - -- F'MACHINE_RADIX
  - -- F'MACHINE_MANTISSA
  - -- F'MACHINE_EMAX
  - -- F'MACHINE_EMIN
  - -- F'MACHINE_OVERFLOWS

- [@DoD 1983: 13.7.3] The values outside the range of safe numbers for real types
- [@DoD 1983: 13.10.1] Any restriction on UNCHECKED_DEALLOCATION
- [@DoD 1983: 13.10.2, F.] Any restriction on UNCHECKED_CONVERSION

[@DoD 1983: Input-Output 14]

- [@DoD 1983: 14., F.] Any implementation-dependent characteristics of the input-output packages

- [@DoD 1983: Implementation-Dependent Features F.]
  - -- [@DoD 1983: F.] The name and type of every implementation-dependent attribute

Outputs: A completed list which characterizes the tool relative to machine dependencies.

15. INFORMATION MANAGEMENT SUPPORT ASSESSORS

These assessors examine the performance, usability, and completeness of the APSE or APSE component functionality related to controlling the information flow during the development of a software system. This includes the organization, accession, modification, dissemination, and processing of any associated information.

15.1 FILE MANAGEMENT CHECKLIST

Purpose: Evaluation of the completeness and power of file management by developing a list of functional capabilities.
[@RM: File Management 7.2.1.3, @RM: Completeness 6.4.9; @RM: Power 6.4.21]

Primary References:
[Peterson 1985] J.L. Peterson and A. Silberschatz, "Operating System Concepts," 2nd edition, Addison-Wesley, 1985.

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 15.1-1) and file manager documentation.

Process: Check off capabilities demonstrated by the file manager or discussed in the documentation.

Outputs: A list of capabilities provided by the file manager.

Table 15.1-1 File Management Capabilities Checklist

FEATURE	FOUND
Operations	
Create	
Read	
Write	
Delete	
Rewind	
Append	
Copy	
Rename	
Update	
Compress	
Expand	
Compare	
Directories	
Operations	
Search	
Create Directory	
Delete Directory	
Rename Directory	
List Directory	
Backup	
Restore	
Structure	
Single-Level (Flat)	
Two-Level	
Tree-Structured (Hierarchical)	
Acyclic Graph	
General Graph	
Storage	
Format	
Record Types	
Fixed Length	
Variable Length	
Byte Count at Beginning	
End of Record Marker	
Blocked Records	
Spanned Records	
Polymorphic Records	
Data	
Text	
ASCII	
EBCDIC	
Numbers	
Integers	
Signed Magnitude	
1's Complement	
2's Complement	
Floating Point	
IEEE Format	
Persistent Knowledge of Ada Types	
Media	
Disk	
Drum	
Magnetic Tape	
Other	
Multi-Volume Files	

Table 15.1-1 File Management Capabilities Checklist (Continued)

FEATURE	FOUND
Allocation Method	
Contiguous	
Linked	
Indexed	
Access Management	
Sequential File	
Direct (Random) Access File	
Primary Indexing	
Secondary Indexing	
Hash-Coded Indexing	
Access Security Protection	
Dynamic Protection Structure	
Data Encryption	
File Password	
Static	
Dynamic	
Multiple	
Access Control	
Controlled Operations	
Read	
Write	
Execute	
Append	
Delete	
Access Matrix	
Global Table	
Access List	
Capability List	
Lock/Key	

15.2 DATABASE MANAGEMENT CHECKLIST

Purpose: Evaluation of the completeness and power of database management by developing a list of functional capabilities.
[@RM: Database (Object) Management 7.2.1.1; @RM: Completeness 6.4.9; @RM: Power 6.4.21]

Primary References: [Martin 1986] D. Martin, "Advanced Database Techniques," MIT Press, 1986.

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 15.2-1) and database manager documentation.

Process: Check off capabilities demonstrated by the database manager or discussed in the documentation.

Outputs: A list of capabilities provided by the database manager.

Table 15.2-1 Database Management Capabilities Checklist

FEATURE	FOUND
Model	
Hierarchical	
Network	
Relational	
Post-Relational	
Object-Oriented	
Data Dictionary Management	
Definitions	
Files, Tables	
Fields, Attributes	
Relationships	
Subschemas	
Views	
Single-Level	
Multilevel	
Objects, Entities	
Data Types	
Subtypes	
User-Defined Types	
Operations	
Implementation Parameters	
Nondatabase Entities	
Listing Descriptions	
Cross-Referencing Descriptions	
Dictionary History	
Automatic Generation of Data Definition Statements	
Data Queries	
Nonprocedural Language (4GL)	
Fill-in-the-Form	
Query by Example	
Report Generation	
Query	
Data Set	
User-Defined	
Update Mode	
Static	
Dynamic	
Access Security Protection	
Access Control	
By View	
By File, Table	
By Object, Entity	
By Relationship	
By Operation (Read, Write, Append)	
Data Encryption	
Dynamic Password	
Keyword Input Protection	
Protection of Stored and Transmitted Data	
Referential Integrity	

Table 15.2-1 Database Management Capabilities Checklist (Continued)

FEATURE	FOUND
Access Conflict and Deadlock Protection	
Access Locking	
Dynamic Backout from Deadlock	
Undoing Multiple Transactions	
Storage	
Full-Length	
Representation with Codes	
Data Packing	
Disk Space Management	
Multivolume Files	
Areas	
File Groups	
File Access Management	
Sequential File	
Direct (Random) Access File	
Primary Indexing	
Secondary Indexing	
Hashing	
Hash-Coded Index	
Database-Key	
Bit-Vector Inverted File (Bit-Index)	
File Linking	
One-to-One Relationship	
One-to-Many Relationship	
Many-to-Many Relationship	
Application Program Interface	
Application Development Language Interface	
Standard DBMS Operations	
Insertion (Record Creation or Addition)	
Modification (Field Update)	
Deletion	
Link Creation and Suppression	
Screen Generators	
Program/Data Independence through Mapping	
Program/Structure Independence Using Multilevel Views	
Backup and Recovery	
Transaction Logging	
Cold Restart	
Warm Restart	
Data Restructuring Capabilities (Views)	
Administration Capabilities	
Interactive	
Dictionary Management	
Access Permission Management	
Data Quality Verification	
Communication Capabilities	
Import, Bulk Data Loading	
Export, Flat File Conversion	
Single System Access	
Multiple System Access	
Distributed Database	
Performance Monitoring/Tuning	
Miscellaneous	
Terminal Independent	
On-Line Help Facility	

15.3 ELECTRONIC MAIL CHECKLIST

Purpose: Evaluation of the completeness and power of electronic mail by developing a list of functional capabilities.
[@RM: Electronic Mail 7.2.1.4; @RM: Completeness 6.4.9; @RM: Power 6.4.21]

Primary References:
[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.
[Classification Schema/E&V Taxonomy Checklists: 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 15.3-1) and mail system documentation.

Process: Check off capabilities demonstrated by the mail system or discussed in the documentation.

Outputs: A list of capabilities provided by the mail system.

Table 15.3-1 Electronic Mail Capabilities Checklist

FEATURE	FOUND
Send	
Receive	
Send Registered Mail	
Immediate Forwarding	
Immediate Reply	
Archive	
Print	
Search For String	
Edit Message To Be Sent	
Read Next Message	
Read Previous Message	
Read First Message	
Read Last Message	
Position To Start Of Message	
Keypad Support	
On-Line Help Facility	
Send To Distribution Lists	
Send Across Network	
Mail Filing	
Configure The Mailbox	
Programmatic Interface	
Message Status	
Incoming — Read/Unread	
Outgoing — Sent/Unsent	

99. OTHER ASSESSORS

This chapter contains instances of E&V technology that do not conveniently fit one of the earlier chapters. It is likely that in future versions of the Guidebook some of these "miscellaneous" instances will be grouped together in new chapters, and therefore moved out of Chapter 99.

99.1 TEXT EDITING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness and power of text editing by developing a list of functional capabilities.
[@RM: Text Editing 7.1.1.1; @RM: Completeness 6.4.9; @RM: Power 6.4.21]

Primary References:
[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.
[Classification Schema/E&V Taxonomy Checklists: 4.4]

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 99.1-1) and text editor documentation.

Process: Check off capabilities demonstrated during editing sessions or discussed in the documentation.

Outputs: A list of capabilities provided by the text editor.

Table 99.1-1 Text Editing Capabilities Checklist

FEATURE	FOUND
Locator Movement	
Left, Right, Up, Down	
Top, Bottom of File	
Next/Previous Word	
Beginning, End of Line	
Beginning, End of Page (Screen)	
Scroll Up, Down, Left, Right	
Page Up, Down, Left, Right	
Search/Replace	
Search Forward, Backward	
Regular Expression Search, Replace	
Multiple Replace	
Buffers	
Copy Text To, From	
Edit Multiple Files	
Split Screen	
Regions	
Set Mark	
Insert Region	
Delete Region	
Copy Region	
Move Region	
Hide, Show Region	
File Manipulation	
Copy From File	
Append To File	
Macros	
Keyboard Macros	
Macro Language	
File Storage	
Save (Continue Editing)	
Quit (No Save)	
Automatic Save	
Versioning (Backup Original, Save Changes Only)	
Baselining	
Miscellaneous	
Terminal Independent	
On-Line Help Facility	
Minimal Redisplay Algorithm (Refresh)	
Key Redefinition	
Undo Command	
Command Recall, Redo	
Command Type-Ahead	
Session Logging	
Spawn Command Language Process	

99.2 LANGUAGE-SENSITIVE EDITING CAPABILITIES CHECKLIST

Purpose: Evaluation of the completeness and power of language-sensitive editing by developing a list of functional capabilities. This list deals only with those features that provide the language-sensitivity to the editor. For a list of features supporting general text editing see the Text Editing Capabilities Checklist [99.1]. This list may be used to evaluate editors which are sensitive to languages such as Ada or FORTRAN as well as word processors which may be viewed as editors which are sensitive to the English language.
[@RM: Text Editing 7.1.1.1, @RM: Completeness 6.4.9, @RM: Power 6.4.21; @RM: Syntax and Semantics Checking 7.3.1.15]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 99.2-1) and text editor documentation.

Process: Check off capabilities demonstrated during editing sessions or discussed in the documentation.

Outputs: A list of capabilities provided by the text editor.

Table 99.2-1 Language-Sensitive Editing Capabilities Checklist

FEATURE	FOUND
Locator Movement	
Next, Previous Word (Identifier, Keyword)	
Beginning, End of Sentence (Statement, Comment)	
Beginning, End of Paragraph (Block)	
Beginning, End of Section (Unit)	
Beginning, End of Document (Compilation)	
Search/Replace	
Word — Where Used, Where Defined	
Sentence	
Paragraph	
Template	
Stub	
Section	
Regions	
Define Region — Word, Sentence, Paragraph, Template, Stub, Section	
Insert Region	
Delete Region	
Copy Region	
Move Region	
Hide, Show Region	
Comment Out Region	
Display	
High-Level Structure	
Unclosed Structures	
Matching Structures	
Permitted Constructs	
Words of Permitted Type	
Miscellaneous	
(User-Defined) Reformat	
On-Line LRM Access	
Support for Mixed Languages	
Analyze Change — Section, Document	
Check Spelling	
Check Grammar (Syntax)	
Check Meaning (Semantics)	
Translate (Compile) — Sentence, Paragraph, Section	
Knowledge-based versus Template-based	
Traceability between Objects	

99.3 PERFORMANCE MONITORING CHECKLIST

Purpose: Evaluation of the completeness of performance monitoring by developing a list of functional capabilities.
[@RM: Performance Monitoring 7.2.1.10, @RM: Completeness 6.4.9]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 99.3-1) and performance monitor documentation.

Process: Check off capabilities demonstrated by the performance monitor or discussed in the documentation.

Outputs: A list of capabilities provided by the performance monitor.

Table 99.3-1 Performance Monitor Capabilities Checklist

FEATURE	FOUND
Hardware	
CPU Time (Real And Virtual)	
Memory Usage	
I/O Channel Traffic	
Terminal Response	
Terminal Connect Time	
Terminal Availability	
Disk Usage	
Disk Space Availability	
Tape Mounts	
Tape Drive Availability	
Printout Quantity	
Software	
Tool Usage	
Program Library Monitoring	
Wall Clock Time	

99.4 COMMAND LANGUAGE INTERPRETER ASSESSMENT QUESTIONNAIRE

Purpose: The document presents a hierarchical breakdown of the command language interpreter shown in Fig. 99.4-1. Requirements for each element in the hierarchy are listed for addressing certain attributes. Each requirement is augmented by one or more questions which address the requirement.
[@RM: Command Language Processing 7.2.3.1, @RM: Attributes 6.]

Primary References:
[E&V Report 1984] "Requirements for Evaluation and Validation of Ada Programming Support Environments, Version 1.0," 17 October 1984, Appendix B of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, November 1984, pp. B-39 - B-44, DTIC Number AD A153 609.
[Requirements for E&V: 4.5]

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire, command language interpreter, and documentation.

Process: Answer the questions by using the command language interpreter, reading the documentation, or asking the vendor of the command language interpreter.

Outputs: Completed questionnaire.

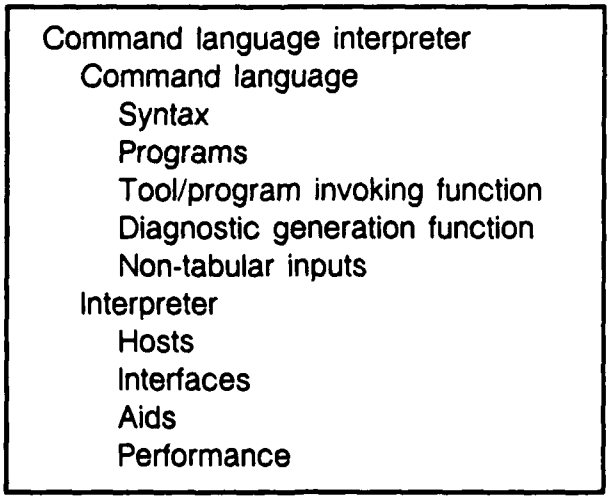

Figure 99.4-1 Command/Language Interpreter Hierarchy

99.5 RADC SOFTWARE QUALITY METRIC WORKSHEETS

Purpose: "The purpose of this guidebook is to provide a comprehensive set of procedures and techniques to enable data collection personnel to apply quality metrics to software products and to evaluate the achieved quality levels." The focus of the RADC report is planning and designing quality into application software throughout the software life cycle. However, many of the questions on the worksheets are equally relevant to systems and support software and may be rephrased to address software that is already in use as opposed to software under development.
[@RM: Attributes 6.]

Primary References: [RADC 1985] T.P. Bowen, G.B. Wigle, and J.T. Tsai, "Specification of Software Quality Attributes Software Quality Evaluation Guidebook," Rome Air Development Center, Griffiss AFB, RADC-TR-85-37, Volume III (of three), February 1985, Appendix A, DTIC Number AD A153 990.

Vendors/Agents: [RADC]

Method: Questionnaire/Worksheet

Inputs: Worksheet, tool, and tool documentation.

Process: Answer the questions by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed worksheet.

99.6 SEI ASSESSMENT OF SOFTWARE ENGINEERING TOOLS

Purpose: The guide provides and discusses a set of standard questions that a potential user may ask about a tool, given that different users will interpret the answers in different ways and attach different degrees of importance to them. The questions are grouped according to the following aspects: 1) Ease of use, 2) Power, 3) Robustness, 4) Functionality, 5) Ease of insertion, and 6) Quality of commercial support. The first four sections are mainly of concern to the actual user of the tool; the last two are of concern to the management of the project that contemplates acquiring the tool.
[@RM: Anomaly Management 6.4.2, @RM: Augmentability 6.4.4,
@RM: Capacity 6.4.6, @RM: Commonality 6.4.7, @RM: Completeness 6.4.9,
@RM: Consistency 6.4.10, @RM: Functional Overlap 6.4.14, @RM: Granularity 6.4.17,
@RM: Maturity 6.4.18, @RM: Operability 6.4.20, @RM: Power 6.4.21,
@RM: Processing Effectiveness 6.4.22, @RM: Proprietary Rights 6.4.23,
@RM: Rehostability 6.4.25, @RM: Simplicity 6.4.29,
@RM: Storage Effectiveness 6.4.31, @RM: System Availability 6.4.32,
@RM: System Compatibility 6.4.34, @RM: Traceability 6.4.35,
@RM: Training 6.4.36, @RM: Visibility 6.4.38]

Primary References: [Firth 1987] R. Firth, V. Mosley, R. Pethia, L. Roberts, W. Wood, "A Guide to the Classification and Assessment of Software Engineering Tools," Software Engineering Institute, Technical Report, CMU/SEI-87-TR-10, August 1987, DTIC Number AD A182 895.

Vendors/Agents: [SEI]

Method: Questionnaire

Inputs: Questionnaire, tool, and tool documentation.

Process: Answer the questions by using the tool, reading the documentation, or asking the vendor of the tool.

Outputs: Completed questionnaire.

99.7 VENDOR EVALUATION QUESTIONNAIRE

Purpose: The purpose of this questionnaire is to provide an overview of the characteristics and policies of a vendor. The questionnaire appears in full in Section 4.8 of the reference cited below. In its current form it applies specifically to Ada compilation system vendors, but most of the questions apply equally well to tool vendors in general. Table 99.7-1 simply lists the titles of the 11 subdivisions of the questionnaire. The questionnaire itself provides 2 to 15 questions under these titles.
[@RM: Cost 6.4.11, @RM: Document Accessibility 6.4.13, @RM: Maturity 6.4.18; @RM: Proprietary Rights 6.4.23, @RM: Training 6.4.36]

Primary References:

[Weiderman 1989] N.H. Weiderman, "Ada Adoption Handbook: Compiler Evaluation and Selection, Version 1.0," Software Engineering Institute, CMU/SEI-89-TR-13, March 1989, DTIC Number AD A207 717.
[Weiderman: Compiler Evaluation and Selection: 4.22]

Vendors/Agents: [SEI]

Method: Questionnaire

Inputs: Blank characterization form and tool documentation.

Process: Gather data and fill in form.

Outputs: Completed vendor characterization form.

Table 99.7-1 Vendor Characterization Form Categories

Corporate structure
Corporate performance
Product lines
Corporate health
Tailoring policies
Support policies
Pricing policies
Runtime policies
Runtime royalties
Source code
Contractual issues
References

99.8 REQUIRED CONFIGURATION QUESTIONNAIRE

Purpose: Assess the required configuration for using a software product. The questionnaire covers the recommended as well as the required configuration since using a product with the minimum required configuration may result in performance which is unacceptable to a user.
[@RM: Required Configuration 6.4.26]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire (see Fig. 99.8-1) and the product documentation.

Process: Answer questions based on the documentation or by asking the product vendor.

Outputs: Completed questionnaire which describes the resources required to run the product.

Host System
What host computer(s) or chip(s) does the product run on?
What operating system(s) (including version and release) does the product run on?
- What is the minimum and recommended configuration of the operating system (parameter settings)?

Main Memory
What is the minimum required RAM to run the product?
- What is the recommended RAM to run the product?

How do the RAM requirements change with the size of the input data file(s)?

Secondary Memory
How much space do the executable(s), object file(s), and runtime system take?
How much space do typical input data file(s) take?
- How does the space vary with the size of the input data file(s)?

How much space do typical output data file(s) take?
- How does the space vary with the size of the output data file(s)?

Peripherals (Disk, Monitor, Printers, Plotters, Mouse, etc.)
What are the required peripherals?
What are the recommended peripherals?

Other Software (APSE, CAIS, Runtime, DBMS, Window Manager, etc.)
What are the other required software products?
What are the other recommended software products?

Figure 99.8-1 Required Configuration Questionnaire

99.9 COST QUESTIONNAIRE

Purpose: Assess the costs of acquiring, running, and supporting a software product. [@RM: Cost 6.4.11]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire (see Fig. 99.9-1) and the product pricing information.

Process: Answer questions based on the pricing information or by asking the product vendor.

Outputs: Completed questionnaire which describes the costs of acquiring, running, and supporting the product.

Product

What is the single copy price?
Are there discounts for volume purchases?
- Does the entire quantity have to be purchased all at once or does the vendor track purchases by organization and automatically apply the discount when the required volume has been reached?

What is the price of a site license?
What is the cost of leasing the product?
- Is the lease perpetual or fixed term?
- If fixed term, what are the renewal terms?

How does the cost depend on characteristics of the host machine?
- What are the costs to add more users or workstations?
- What are the costs to move to a different host machine?

Are there discounts for government organizations?
Are there discounts for academic institutions?
Can the product be returned during some specified period for a full refund?
If available, what is the cost of purchasing the source code?

Maintenance

What is the one year maintenance cost?
- What does it provide in terms of:
  - Support?
  - Bug Fixes?
  - Upgrades?
  - Documentation Updates?

What happens in the event maintenance has been dropped?
What is the cost of hot line support?
  - Is there an 800 number?

If there is an electronic bulletin board for problem reports and resolutions, what is the subscription fee?
What are the costs of making application-specific changes?

Training

What is the cost of:
- On-site training?
- Seminars?
- Video training courses?
- Computer based training?
- Users' group membership?
- Newsletter subscription?
- In-house consultants?

Are there training credits offered with the purchase of the product?
- How much are they worth?
- What can they be used for?
- How long are they good for?

Documentation

What is the price of:
- Installation guide?
- Users' guide?
- Reference manual?
- Interface manual?
- Quick reference card?
- Keyboard template?

Miscellaneous

If the product requires the purchase of other hardware or software, what does that cost?
What is the cost of installing the product?
What is the cost of running the product?
- Cost per run or session?
- Runs or sessions per day, week, month, or year?

What is the cost of supporting the product?
- Cost of computer resources?
- Cost of operations or support personnel?

Figure 99.9-1 Cost Questionnaire

99.10 MATURITY QUESTIONNAIRE

Purpose: Assess the maturity of a software product.
[@RM: Maturity 6.4.18]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire (see Fig. 99.10-1) and the product historical information.

Process: Answer questions based on the historical information or by asking the product vendor.

Outputs: Completed questionnaire which describes the maturity of the product.

Product History
When was the product first released?
What is the current version and release?
What is the frequency of new versions and releases?
What are the procedures for testing new versions and releases?
- Alpha testing?
- Beta testing?

User Community
How many active users of the product?
- Will the vendor supply references?
- What applications has the product been used on?

Is the vendor willing to share problem reports and resolutions with the users?
Is there a users' group?
- How often do they meet?

Product Evaluation
How has the product been rated in the literature?
Are there independent evaluations of the product available?

Figure 99.10-1 Maturity Questionnaire

99.11 LICENSING ISSUES QUESTIONNAIRE

Purpose: Assess the licensing agreement for a software product.
[@RM: Proprietary Rights 6.4.23]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire (see Fig. 99.11-1) and the product licensing agreement.

Process: Answer questions based on the licensing agreement or by asking the product supplier.

Outputs: Completed questionnaire which describes the licensing agreement for a product.

What, specifically, is being purchased, leased, or otherwise acquired?
- Hardware?
- Software?
- Utilities?
- Documentation?
- If lease, is it perpetual or fixed term?
  - If fixed term, what are the renewal terms?

Is any part of the product "free" or shareware?
- What are the restrictions and/or obligations in using the product?

What, specifically, is covered by patents, copyrights, trademarks, or agreements?
- Is any of the product copy protected?
- Can the user make backup copies for protection?
- Can the documentation be copied?

Are there limitations on the number of users or workstations using the product?

Can the product be moved to a different machine or does it have to stay on a specific machine?

Is the source code available?
- If so, what are the licensing terms?
- If not, can it be put into escrow to protect the user in the event that the supplier goes out of business?
- Who holds the rights to the user-modified source code?
  - What is the effect on the maintenance agreement?

What is the supplier's obligation to correct deficiencies?

What is the supplier's obligation to maintain upward compatibility across new versions or releases?

What is the supplier's obligation to provide interface information to the user?
- What is the supplier's obligation to maintain fixed syntax and semantics of the product's interfaces?

What are the supplier's rights to:
- Runtime versions of the product (no development tools)?
- Objects generated by the product?
- What are the procedures to account for and collect the royalties?

What are the user's rights to the executables and/or source code in the event that:
- The supplier goes out of business?
- The supplier is bought out by another company?
- The supplier discontinues the product?
- The supplier issues a new version or release?

What obligations of the supplier to third parties are inherited by the user?

What exceptional conditions and penalty clauses are in the agreement?

Figure 99.11-1 Licensing Issues Questionnaire

99.12 SOFTWARE PRODUCTION VEHICLE(S) QUESTIONNAIRE

Purpose: Assess the software production vehicle(s) of a software product.
[@RM: Software Production Vehicle(s) 6.4.30]

Primary References:

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Questionnaire (see Fig. 99.12-1) and the product specification data.

Process: Answer questions based on the specifications or by asking the product vendor.

Outputs: Completed questionnaire which describes the software production vehicle(s) of the product.

Requirements/Design

What front end methodologies are used to develop the product?
What front end software tools are used to develop the product?

Coding

What source languages are used to implement the product?
- Are the versions based on standards?

Is the source code automatically generated from the front end tools?
- What code generator(s) are used?

What compiler (assembler, interpreter) version is used?
- What parameters are set during compilation?

What is the host computer and operating system (including models and versions)?
- Is the development platform the same as the product host?
  - If not, what is the development computer and operating system?

What linker version is used?
- What parameters are set during linking?

Testing and Evaluation

What testing techniques are used?
- Static analysis?
- Dynamic analysis:
  - Structural coverage?
  - Domain/Path coverage?
  - Symbolic testing?
  - Mutation analysis?
  - Functional testing?
  - Random testing?

What release testing procedures are used?
- Alpha testing?
- Beta testing?

What testing tools are used?

User Feedback

What mechanisms are there for collecting user feedback?
What mechanisms are there for evaluating user feedback?
What mechanisms are there for responding to user feedback?

Configuration Management

What procedures are there for managing the configuration of the product?
What tools are there for managing the configuration of the product?

Figure 99.12-1 Software Production Vehicle(s) Questionnaire

APPENDIX A
CITATIONS

[ACEC 1986] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) Reader's Guide," Air Force Wright Aeronautical Laboratories, Document Number D500-11790-2, 10 August 1988, DTIC Number AD B125 147.

[ACVC 1989] Ada Compiler Validation Procedures, Version 2.0, AJPO, May 1989.

[ALS 1984] SofTech, "Ada Language System (ALS) Specification," CR-CP-0059-A00, August 1984.

[ARTEWG 1987] "Catalogue of Ada Runtime Implementation Dependencies," Association for Computing Machinery, Special Interest Group on Ada, Ada Runtime Environment Working Group, 1 December 1987.

[ARTEWG 1988] "A Framework for Describing Ada Runtime Environments," Proposed by Ada Runtime Environment Working Group (SIGAda), Ada Letters, Volume VIII, Number 3, May/June 1988, pp. 51-68.

[Barnes 1985] *Proceedings of the International Ada Conference*, Paris, eds. J.G.P. Barnes and G.A. Fisher, Jr, Cambridge University Press, 1985.

[Barstow 1981] D.R. Barstow and H.E. Shrobe, "Observations on Interactive Programming Environments," [@Wassermann 1981], pp. 286-301, 1981.

[Buxton 1980] [@DoD 1980].

[CAIS] [@DoD 1986].

[CAIS-A] [@MIL 1989].

[CIVC 1989] "CIVC1 Implementor's Guide," Wright Research and Development Center, CIVC-FINL-019, October 1989, in progress.

[DACS 1979] The DACS Glossary, A Bibliography of Software Engineering Terms, October 1979.

[DeMillo 1986] R.A. DeMillo, "Functional Capabilities of a Test and Evaluation Subenvironment in an Advanced Software Engineering Environment," Georgia Institute of Technology GIT-SERC-86/07, 20 October 1986.

[DoD APSE Analysis 1984] [@E&V Report: DoD APSE Analysis Report C.]

[DoD 1977] DoD, "Requirements for High Order Computer Languages (IRONMAN), U.S. Department of Defense, 1977, DTIC Number AD A100 403.

[DoD 1980] J.N. Buxton, "Requirements for Ada Programming Support Environments — STONEMAN," U.S. Department of Defense, February 1980, DTIC Number AD A100 404.

[DoD 1982] "Software Development Methodologies and Ada (METHODMAN)," U.S. Department of Defense, 1982.

[DoD 1983] ANSI/MIL-STD-1815A-1983, Reference Manual for the Ada Programming Language, U.S. Department of Defense, 17 February 1983, DTIC Number AD A131 511.

[DoD 1986] DoD-STD-1838, Common APSE Interface Set (CAIS), U.S. Department of Defense, 9 October 1986, DTIC Number AD A157 589.

[DoD Trusted Computer Report 1983] "Trusted Computer System Evaluation Criteria," CSC-STD-001-83, U.S. Department of Defense Computer Security Center, 15 August 1983, DTIC Number AD A207 905.

[Donaldson 1988] C. Donaldson and P.B. Dyson, "Computer-Aided Systems and Software Engineering Products for Time-Critical Applications Development," Software Productivity Solutions (SPS), Inc., April 1988.

[E&V Plan] [@E&V Report 1984: E&V Plan A]. [@E&V Report 1987: E&V Plan A].

[E&V Reference Manual] [@RM].

[E&V Report 1984] Evaluation and Validation (E&V) Team Public Report, Volume I, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, 30 November 1984, DTIC Number AD A153 609.

[E&V Report 1985] "Evaluation and Validation (E&V) Team Public Report," Volume II, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, November 1985, DTIC Number AD A172 343.

[E&V Report 1987] "Evaluation and Validation (E&V) Team Public Report," Volume III, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, September 1987, DTIC Number AD A196 164.

[E&V Requirements 1984] [@E&V Report 1984: E&V Requirements B.].

[E&V Requirements 1987] [@E&V Report 1987: E&V Requirements D].

[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[E&V Tools and Aids 1987] [@E&V Report 1987: Tools and Aids C].

[Firth 1987] R. Firth, V. Mosley, R. Pethia, L. Roberts, W. Wood, "A Guide to the Classification and Assessment of Software Engineering Tools," Software Engineering Institute, Technical Report, CMU/SEI-87-TR-10, August 1987, DTIC Number AD A182 895.

[Gray 1987] L. Gray, "Using the SEI's Methodology for Evaluating Ada Environments: A Comparison of VAX/VMS to Rational," Proceedings of the AIAA Computers in Aerospace VI Conference, 7-9 October 1987.

[Grund 1985] E.C. Grund, L.A. Hilliard, and K.A. Younger, "Key Characteristics of Ada Programming Support Environments," MITRE Corporation, ESD-TR-85-144, 9590, July 1985, DTIC Number AD B096 137.

[Henderson 1987] P.B. Henderson and D. Notkin, "Integrated Design and Programming Environment," Computer, IEEE, November 1987.

[Hogan 1985] M.O. Hogan and S.M. Prud'homme, "Definition of a Production Quality Compiler," Aerospace Corporation, Technical Report, July 1985, DTIC Number AD A182 445.

[Houghton 1983] R.C. Houghton, Jr., "A Taxonomy of Tool Features for the Ada Programming Support Environment (APSE)," U.S. Department of Commerce, National Bureau of Standards, NBSIR-81-2625, December 1982, Issued February 1983.

[Houghton 1987] R.C. Houghton, Jr. and D.R. Wallace, "Characteristics and Functions of Software Engineering Environments: An Overview," ACM Software Engineering Notes, Vol. 12, No. 1, January 1987.

[Howden 1982] W.E. Howden, "Contemporary Software Development Environments," Communications of the ACM 25(5), pp. 318-329, 1982.

[IDA 1985] A.A. Hook, G.A. Riccardi, M. Vilot, and S. Welke, "User's Manual for the Prototype Ada Compiler Evaluation Capability (ACEC)," Version 1, Institute for Defense Analysis, IDA Paper P-1879, October 1985, DTIC Number AD A163 272.

[ISTAR 1987] Workshop on Future Development Environments, Information Science and Technology Assessment for Research, Conference on Information Mission Area (IMA) Productivity, Department of Army Director of Information Systems for Command, Control, Communications and Computers, 13-15 April 1987, pp 28.

[Jackson 1985] A.R. Jackson, "Abstract Data Types and the IPSE Database," [@McDermid 1985], pp. 135-145, 1985.

[Kean 1985] E.S. Kean and F.S. Lamonica, "A Taxonomy Of Tool Features For A Life Cycle Software Engineering Environment," Rome Air Development Center, Griffiss AFB, June 1985, DTIC Number AD B096 355.

[Lawlis 1989] P.K. Lawlis, "Supporting Selection Decisions Based on the Technical Evaluation of Ada Environments and Their Components," PhD. dissertation, Arizona State University, August 1989.

[Lehman 1981] M.M. Lehman, "The Environment of Program Development, Maintenance Programming, and Program Support," [@Wasserman 1981], pp. 3-14, 1981.

[Long 1988] F.W. Long, and M.D. Todd, "Evaluating Tool Support Interfaces," Ada in Industry, Proceedings of the Ada-Europe Conference, Munich, 7-9 June 1988, Cambridge University Press, 1988.

[Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986.

[Martin 1986] D. Martin, "Advanced Database Techniques," MIT Press, 1986.

[McDermid 1984] J. McDermid and K. Ripken, "Life Cycle Support in the Ada Environment," Cambridge University Press, 1984.

[METHODMAN] [@DoD 1982].

[Mich 1986] R.M. Clapp, L. Duchesneau, R.A. Volz, T.N. Mudge, and T. Schultze, "Toward Real-Time Performance Benchmarks for Ada," Electrical Engineering and Computer Science Dept., Univ. of Michigan, RSD-TR-6-86, January 1986, pp. 1-25.

[MIL 1989] "Common APSE Interface Set, Revision A," MIL-STD-1838A, U.S. Department of Defense, April 1989, DTIC Number AD A157 589.

[MITRE BGT 1986] S.R. Rainer and T.P. Reagan, "User's Manual for the Ada Compilation Benchmark Generator Tool," MITRE Corp. MTR-87W00192-01, January 1988.

[Morton 1985] R.P. Morton and J.C. Wileden, "Information Interface Related Sources," Institute for Defense Analyses, SEE-INFO-003-001.0, IDA Paper p-1821, April 1985 (Appendix 2, L. Stucki, "Some Thoughts on a Taxonomy for Software Engineering Objects"), DTIC Number AD A185 664.

[NBS Taxonomy] [@Houghton 1983].

[Nissen 1984] J.C.D. Nissen, B.A. Wichman, et al.,"Guidelines for Ada Compiler Specification and Selection," in Ada: *Language, Compilers And Bibliography*, ed. M.W. Rogers, Cambridge University Press, 1984.

[Notkin 1981] D.S. Notkin and A.N. Habermann, "Software Development Environment Issues as Related to Ada," [@Wasserman 1981], pp. 107-133, 1981.

[Oberndorf 1988] P.A. Oberndorf, "The Common Ada Programming Support Environment (APSE) Interface Set (CAIS)," IEEE Transactions on Software Engineering, Vol. 14, No. 6, June 1988.

[Peterson 1985] J.L. Peterson and A. Silberschatz, "Operating System Concepts," 2nd edition, Addision-Wesley, 1985.

[RADC 1985] T.P. Bowen, G.B. Wigle, and J.T. Tsai, "Specification of Software Quality Attributes Software Quality Evaluation Guidebook," Rome Air Development Center,

Griffiss AFB, RADC-TR-85-37, Volume III (of three), February 1985, DTIC Number AD A153 990.

[RM] "Evaluation and Validation (E&V) Reference Manual," Version 2.0, Wright Research and Development Center, WRDC TR-89-?, Wright Patterson AFB, September 1989, DTIC Number pending.

[SEE Taxonomy] [@Kean 1985].

[STARS-SEE 1985] "Proposed Version 001.0," STARS Joint Service Team for Software Engineering Environments, Stars Joint Program Office, October 1985.

[Stenning 1981] V. Stenning, T. Froggart, R. Gilbert, and E. Thomas, "The Ada Environment: A Perspective," [@Wasserman 1981], pp. 36-46, 1981.

[STONEMAN] [@DoD 1980].

[Stuebing 1988] H.G. Stuebing, "Evaluation of Computer Aided Systems/Software Engineering Products for Time-Critical Naval Systems," Proceedings of the Conference *Methodologies and Tools for Real Time Systems*, November 14-15, 1988.

[Texas Instruments 1985] The APSE Interactive Monitor, Texas Instruments, Slide Presentation to the E&V Team, 5 September 1985.

[UK AES 1986] R.H. Pierce, I. Marshall, and S.D. Blude, "An Introduction to the MoD Ada Evaluation System," Software Sciences Ltd., Report Number 5485, June 1986.

[Wasserman 1981] A.I. Wasserman, *Tutorial: Software Engineering Environments*, IEEE, 1981.

[Weiderman 1987] N. Weiderman and N. Haberman, "Evaluation of Ada Environments," Software Engineering Institute, Technical Report CMU/SEI-87-TR-1, March 1987, DTIC Number AD A180 905.

[Weiderman 1987b] N.H. Weiderman, et al., "Ada for Embedded Systems: Issues and Questions," Software Engineering Institute, Technical Report CMU/SEI-87-TR-26, December 1987, DTIC Number AD A191 096.

[Weiderman 1989] N.H. Weiderman, "Ada Adoption Handbook: Compiler Evaluation and Selection, Version 1.0," Software Engineering Institute, Technical Report CMU/SEI-89-TR-13, March 1989, DTIC Number AD A207 717.

[WIS CEC 1985] G. Gicca and C. Stacey, "Component Evaluation Criteria," GTE Government Systems, Technical Report, 16 August 1985.

[WIS CEG 1985] "WIS Compiler Evaluation Guidelines," GTE Labs, Technical Report, 1985.

APPENDIX B
ACRONYMS AND ABBREVIATIONS

ACEC	Ada Compiler Evaluation Capability
ACM	Association for Computing Machinery
ACVC	Ada Compiler Validation Capability
AES	Ada Evaluation System
AFB	Air Force Base
AFWAL	Air Force Wright Aeronautical Laboratories
AIE	Ada Integrated Environment
AJPO	Ada Joint Program Office
ALS	Ada Language System
ALS/N	Ada Language System/Navy
ANNA	Annotation Language for Ada
ANSI	American National Standards Institute
APSE	Ada Programming Support Environment
ARTEWG	Ada RunTime Environment Working Group (SIGAda)
ASCII	American Standard Code for Information Interchange
ATF	Advanced Tactical Fighter
AVF	Ada Validation Facility
AVO	Ada Validation Organization
BGT	Benchmark Generator Tool
BSI	British Standards Institute (UK)
CAIS	Common APSE Interface Set
CIVC	CAIS Implementation Validation Capability
CLI	Command Language Instruction
CMU	Carnegie Mellon University
CORE	Controlled Requirements Expression
CPU	Central Processing Unit
CSC	Computer Software Component
DACS	Data and Analysis Center for Software
DoD	Department of Defense
DTIC	Defense Technical Information Center
EBCDIC	Extended Binary Coded Decimal Interchange Code
ESD	Electronic Systems Division (Air Force)
E&V	Evaluation and Validation
GB	Guidebook
GIT	Georgia Institute of Technology
GKS	Graphical Kernel System
IBM	International Business Machines Corporation
IDA	Institute for Defense Analysis

IEEE	Institute of Electrical and Electronics Engineers, Inc.
IMA	Information Mission Area
IPSE	Integrated Project Support Environment
ISTAR	Information Science and Technology Assessment for Research
I/O	Input/Output
JIAWG	Joint Integrated Avionics Working Group
KAPSE	Kernel Ada Programming Support Environment
KIT	KAPSE Interface Team
KITIA	KAPSE Interface Team for Industry and Academia
LHX	Light Helicopter Experimental
LRM	Language Reference Manual [@DoD 1983]
MAPSE	Minimal Ada Programming Support Environment
MCCS	Mission-Critical Computer System
MIT	Massachusetts Institute of Technology
MMI	Man-Machine Interface
MoD	Ministry of Defense (UK)
NADC	Naval Air Development Center
NBS	National Bureau of Standards
NTIS	National Technical Information Service
OCD	Operational Concept Document
OS	Operating System
PCTE	Portable Common Tool Environment
PDL	Program Design Language
PIWG	Performance Issues Working Group (SIGAda)
RAM	Ramdom Access Memory
RM	Reference Manual
SDI	Strategic Defense Initiative
SEE	Software Engineering Environment
SEI	Software Engineering Institute
SIGAda	Special Interest Group for Ada of the Association for Computing Machinery (ACM)
SPS	Software Productivity Solutions
SQL	Structured Query Language
STARS	Software Technology for Adaptable, Reliable Systems
TASC	The Analytic Sciences Corporation
TOR	Technical Operating Report
UK	United Kingdom
VAX	Virtual Address Extension
VMS	Virtual Memory System
VSR	Validation Summary Report
V&V	Verification and Validation
WIS	WWMCCS Information System
WRDC	Wright Research and Development Center, formerly AFWAL
WWMCCS	WorldWide Military Command and Control System

APPENDIX C
FORMAL GRAMMAR

This appendix specifies sections of the Reference Manual and Guidebook (Reference System) as a formal grammar. The sections include chapters four through seven of the Reference Manual (RM), chapters four through 99 of the Guidebook (GB), all explicit references, the tables of contents, the indices, and the citations. The specification is presented as a partitioned grammar for convenience.

(The grammar is presented in a modified Backus-Naur form. Brackets represent optionality when alone, and may be marked by an asterisk "*" to denote 0-N instances of the production, or by a sharp "#" to denote 1-N instances. Angle brackets denote comments in place of productions which are too elaborate to express here. All terminals of the grammar are expressed as quoted literals, or composite literals based on characters and character strings.)

C.1 FORMAL REFERENCES

Throughout the Reference System, whenever formal references are made, a single consistent set of grammar rules are used. This includes reference from one volume to the other, reference from one section in a volume to another section in the same document, and reference to documents outside the Reference System.

```
reference_list  ::= "[" references   [";"  references   ]* "]"

references      ::= reference [ "," reference ]*

reference       ::= "@" phrase [ ":" [ phrase ]
                    [ designator_list ]]
                    [ phrase ] designator_list

phrase_list     ::= phrase [ "," phrase ]*
```

phrase ::= <text lacking special characters>

designator_list ::= designator ["," designator]*

designator ::= [lead "."] lead ["." digits]*

lead ::= [digits] caps

digits ::= one_to_nine [zero_to_nine]*

one_to_nine ::= ("1"—"9")

zero_to_nine ::= ("0"—"9")

caps ::= ("A"—"Z")

C.2 FORMAL CHAPTERS

The formal chapters of the Reference System are defined here.

C.2.1 Chapter Components

The following rules define the components which are used to compose formal chapter entries.

header ::= designator phrase

prolog ::= header purpose primary [host]
[vendors_agents]

purpose ::= "Purpose:" text

host ::= "Host/OS:" text

primary ::= "Primary References:" reference_list

vendors_agents ::= "Vendors/Agents:" reference_list

method ::= meth_description inputs
process outputs

meth_description ::= "Method:" text

inputs ::= "Inputs:" text

process ::= "Process:" text

outputs ::= "Outputs:" text

citations ::= "Citations:" [citations]#

synopsis_text ::= "Synopsis:" text

methods ::= "Methods:" reference_list

text ::= < prose text >

C.2.2 Chapter Entries

Each numbered section of the formal chapters follows a specific grammar rule. The following rules define the format of each chapter entry.

synopsi ::= header citations synopsis_text [methods]

E&V_technology ::= prolog method

C.2.3 Formal Chapter Ordering

The formal portion of the GB is found in Chapters 4 through 99.

formal_chapters ::= [synopsis]*

[E&V_technology]*

.

.

.

[E&V_technology]*

C.3 TABLE OF CONTENTS

The table of contents shares some features with the rest of the formal aspects of the GB.

table_of_contents ::= [chapter]* index

chapter ::= designator phrase digits

C.4 CITATIONS

The citations are found in Appendix A, and have a formal structure as defined in the following grammar. The (semantic) form of citation text is taken from the standard for IEEE Software Magazine.

citations ::= [citation]*

citation ::= *key body* "."

key ::= "[" phrase_list "]"

body ::= [reference_list] phrase_list

APPENDIX D
VENDORS AND AGENTS

[ARTEWG]
Mike Kamrad (612) 456-7315
Unisys Computer Systems Division
MS/Y41A6
P.O. Box 64525
St. Paul, MN 55164-0525

[BSI, Milton-Keynes, UK] 0908-220908 x2313
J.B. Souter
BSI Quality Assurance
P.O. Box 375
Milton Keynes MK14 6LL
UK

[Cambridge University Press]
Cambridge University Press
32 East 57th Street
New York, NY 10022

[Defense Technical Information Center]
Defense Technical Information Center (703) 274-7633
Cameron Station
Alexandria, VA 22314

[DACS]
Data & Analysis Center for Software (315) 336-0937
RADC/COED
Bldg 101
Griffiss AFB, NY 13441-5700
Attn: Document Ordering

[E&V Team]
Mr. Raymond Szymanski (513) 255-2446
WRDC/AAAF -6730
Wright-Patterson AFB AV 785-2446
OH 45433-6543 -6730
Milnet: szymansk@ajpo.sei.cmu.edu

[GIT]
Georgia Institute of Technology (404) 894-3180
Software Engineering Research Center
Atlanta, GA 30332-0280

[MITRE]
MITRE Corporation (703) 883-6000
Civil Systems Division
7525 Colshire Drive
McLean, VA 22102-3481

[National Technical Information Service]
National Technical Information Service (703) 487-4650
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

[PIWG]
Dan M. Roy (301) 464-6800
Ford Aerospace
7401-D Forbes Blvd.
Seabrook, MD 20706

[RADC]
Rome Air Development Center (COEE) (315) 330-4654
Griffiss AFB, NY 13441-5700

[SEI]
Software Engineering Institute (412) 268-7700
Carnegie Mellon University
Pittsburgh, PA 15213

[SofTech, Inc.]
Teresa L. Banks (513) 429-3241
SofTech, Inc.
3100 Presidential Drive
Fairborn, OH 45324-2039
ARPAnet: hillm@wpafb-jalcf

[SPS]
Software Productivity Solutions, Inc. (407) 984-3370
P.O. Box 361697
Melbourne, FL 32936

[UMich]
Russel M. Clapp, Louis Duchesneau, Richard A. Volz, (313) 764-1817
Trevor N. Mudge, and Timothy Schultze
The Robotics Research Laboratory
The University of Michigan
Ann Arbor, MI 48109